**VŠB – Technická univerzita Ostrava**
**Fakulta elektrotechniky a informatiky**
**Katedra informatiky**

# Systém pro tvorbu hudebního doprovodu pomocí technologie Kinect

# System for Creation of Music Accompaniment by Kinect Technology

**2015** **Marcel Graňák**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Bc. Marcel Graňák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Systém pro tvorbu hudebního doprovodu pomocí technologie Kinect
System for Creation of Music Accompaniment by Kinect Technology

Zásady pro vypracování:

Cílem diplomové práce je pomocí pohybového senzoru Kinect vytvořit speciální systém sloužící jako hudební doprovod, kde jednotlivé osoby stojící před pohybovým senzorem reprezentují hudební nástroje. Systém bude tvořit sada knihoven, které budou komunikovat pomocí technologie WCF a umožní tak modulární připojování více senzorů Kinect.

Jednotlivé cíle práce jsou:
1. Prostudovat možnosti pohybového senzoru Kinect.
2. Navrhnout komunikační architekturu založenou na WCF pro jednotlivé části výsledného systému.
3. Implementace a návrh systému.
4. Navrhnout uživatelské rozhraní a proces nastavování a výběru nástrojů.
5. Provést experimenty, které budou demonstrovat výsledný hudební doprovod a také ověřit přenosové rychlosti a zpoždění mezi pohybem a cílovým zvukem.

Seznam doporučené odborné literatury:

[1] Jarrett Webb, James Ashley: Beginning Kinect Programming with the Microsoft Kinect SDK,Apress; 1 edition (February 23, 2012), ISBN-10: 1430241047
[2] Kinect for Windows - Developer SDK, http://www.microsoft.com/en-us/kinectforwindows/develop/

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015


doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 7.5.2015

Podpis .....

## Poďakovanie

Rád by som sa poďakoval rodičom, rodine a priateľom za pomoc a podporu, a taktiež aj vedúcemu práce Ing. Janovi Martinovičovi, Ph.D, za odbornú pomoc a konzultácie pri realizácii tejto diplomovej práce.

# Abstrakt

Cieľom práce je overenie možností pohybového senzora Kinect a následne vytvorenie systému na tvorbu hudobného sprievodu. Systém pozostáva z oddelených častí, ktoré medzi sebou komunikujú pomocou technológie WCF. V práci sú popísané možnosti a vlastnosti senzora a ostatných použitých technológií. V druhej časti práce je následne popísaný návrh a realizácia vývoja systému a jeho jednotlivých častí. Na konci práce sú uvedené aj výsledky testov komunikácie v rámci celého systému za rôznych podmienok.



# Kľúčové slová

Kinect, WCF, WPF, publish-subscribe, hudobný sprievod, gestá



# Abstract

The aim of this diploma thesis is to verify possibilities of the Kinect sensor and to create system for music accompaniment. Whole system is made of separated parts communicating to each other by using WCF technology. The possibilities and characteristics of the sensor and other used technologies are written in this thesis. Second part of thesis contains design and implementation of all parts of the system. Results of communications tests of system parts, are at the end of this thesis, by using different conditions.



# Keywords

Kinect, WCF, WPF, publish-subscribe, musical accompaniment, gestures

# Zoznam skratiek v dokumente

| Skratka | Význam |
|---|---|
| **API** | Application Programming Interface |
| **HTTP** | Hypertext Transfer Protocol |
| **MSMQ** | Microsoft Message Queuing |
| **MSR** | Microsoft Research |
| **NUI** | Natural User Interface |
| **OS** | Operating System |
| **RGB** | RGB color model (red, green, blue) |
| **SOAP** | Simple Object Access Protocol |
| **WCF** | Windows Communication Foundation |
| **WPF** | Windows Presentation Foundation |
| **XML** | Extensible Markup Language |
| **XAML** | Extensible Application Markup Language |

# Obrázky v dokumente

# Obsah

# Úvod

Vývoj informačných technológií sa posúva neustále dopredu. Doba prvých počítačov, ktoré zaberali niekoľko miestností a vykonávali len triviálne operácie je už dávno za nami. Zariadenia, ktoré sa používajú dnes majú mnohonásobne väčší výkon, vykonávajú nespočetné množstvo zložitých operácií a ich použitie nie je vo svojej podstate ničím limitované.

Tak ako sa mení a zlepšuje výkon moderných technológií, tak sa musí meniť aj spôsob akým sa daná technológia ovláda. Spôsoby ovládania akými sú dotykové, hlasové či pohybové ovládanie sú neustále vo vývoji a prinášajú nové možnosti ale aj problémy. Pohybové ovládanie, ktorému sa v práci budem venovať predstavuje nový spôsob interakcie človeka s počítačom, či konkrétnou aplikáciou. V roku 2010 si americká spoločnosť Microsoft podala žiadosť o patent na zariadenie na sledovanie a detekciu pohybu osôb (popis patentu sa nachádza tu [1]). Z projektu „Natal“ vznikol produkt s oficiálnym názvom „Kinect“ (viac o oficiálnom predstavení produktu [1]), ktorý sa pôvodne stal pohybovým ovládaním pre hernú konzolu Xbox. Neskôr sa jeho použitie rozšírilo aj do iných oblastí.

V prvej kapitole tejto práce najskôr predstavím pohybový senzor Kinect a jeho možnosti pre tvorbu aplikácií. Tieto možnosti aplikujem na tvorbu systému, ktorý bude slúžiť na rozvíjanie motorických schopností detí za pomoci vytvárania hudobného sprievodu. Systém bude pozostávať z rôznych technológií, ktorých možnosti priblížim v druhej kapitole. V nasledujúcej časti práce si rozoberieme návrh a implementáciu výsledného systému. Vytvorený systém budem následne testovať z hľadiska časovej náročnosti na prenos potrebných dát, detekcie konkrétnych gest a nadefinovaných akcií. Výsledky testovania si priblížime v štvrtej kapitole.

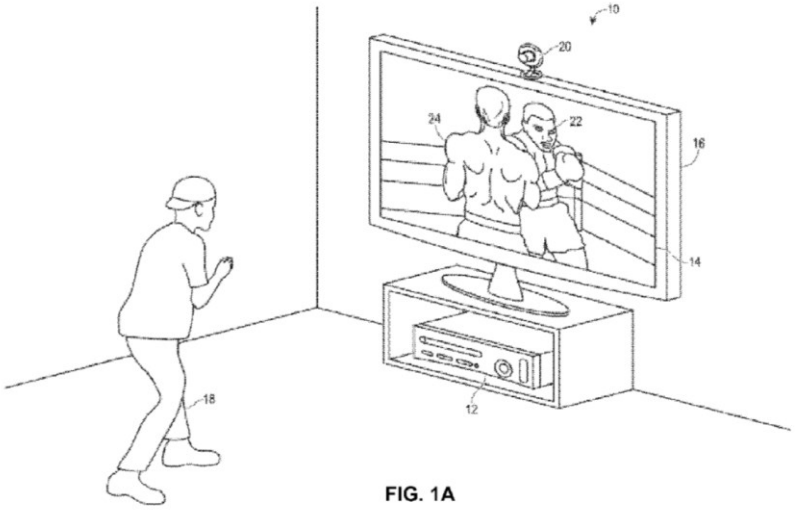
# 1 Kinect

Zariadenie vyvinuté spoločnosťou Microsoft predstavuje nový spôsob interakcie s aplikáciami. V tejto kapitole si predstavíme jeho vývoj, technológiu a priblížime si jeho možnosti.

## 1.1 História

Výskum rozpoznávania pohybu začal približne v 70-tych rokoch. V roku 1979 Chris Schmandt spoločne s MIT Media Lab demonštroval kľúčový koncept budúceho zariadenia Kinect a to detekciu gest a rozpoznávanie reči. V 90-tych rokoch IBM Research a Mark Lucente predstavili projekt rozpoznávania pohybu a slov nazvaný DreamSpace. Viac informácií o histórii je možné získať v prvej kapitole použitej literatúry [2].

Prvý koncept pre projekt „Natal“ (kódové označenie pre Kinect) bol predstavený 1. júna 2009. Natal bol demonštrovaný na konferencii Microsoft E3 2009 na projektoch ako hra „Burnout Paradise“ či projekt „Milo and Kate“, informácie o ďalších projektoch je možné nájsť tu [4]. Projekt Natal predstavil simultánne snímanie štyroch osôb či detekciu prstov. Tieto demonštrácie ukázali veľký potenciál senzoru pre použitie primárne v hernom odvetví.




*Obrázok 1.1: Ukážka registrácie patentu pohybového senzoru od Microsoftu (zdroj: [1])*

Spoločnosť Microsoft, si v roku 2010 podala patent na senzor (Obrázok 1.1) a neskôr predstavila projekt Natal ako Kinect 13. júna 2010 na konferencii E3 2010. Názov Kinect sa skladá z anglických slov „kinetic“ (kinetický, pohybový) a „connection“ (spojenie, pripojenie), čo popisuje kľúčové aspekty zariadenia. Dňa 4. novembra 2010 bol v severnej Amerike spustený predaj zariadenia Kinect, spoločne s novou hernou konzolou Xbox 360, pre ktorú je Kinect primárne určený.

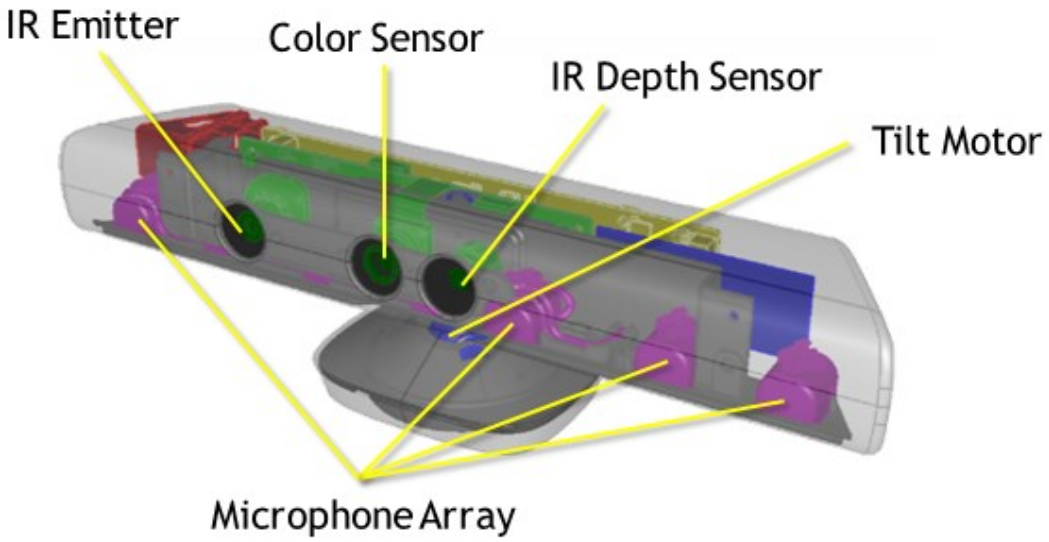
Prvé zariadenie bolo určené výlučne pre herný priemysel. Preto 1. februára 2012 Microsoft predstavil komerčnú verziu „Kinect for Windows“, spoločne s vydaním SDK pre vývoj aplikácií.

## 1.2 Technológia

Na ovládanie zariadení, počítačov a aplikácií je potrebné používať rôzne druhy rozhraní. Od prvotných rozhraní, akými boli rôzne ovládacie prvky (tlačidlá, páky, ovládacie panely a iné), klávesnice až po prvé intuitívne zariadenia akými je napríklad počítačová myš. Ďalšími intuitívnymi rozhraniami sa neskôr stali dotykové ovládanie, hlasové ovládanie či neskôr pohybové ovládanie. Tieto rozhrania získali pomenovanie Natural User Interface (NUI). NUI je pojmom, ktorý popisuje priamy a intuitívny spôsob ovládania, bez nutnosti učiť sa spôsob manipulácie s rozhraním, viac o NUI [5]. Do tejto skupiny patrí aj samotný Kinect.

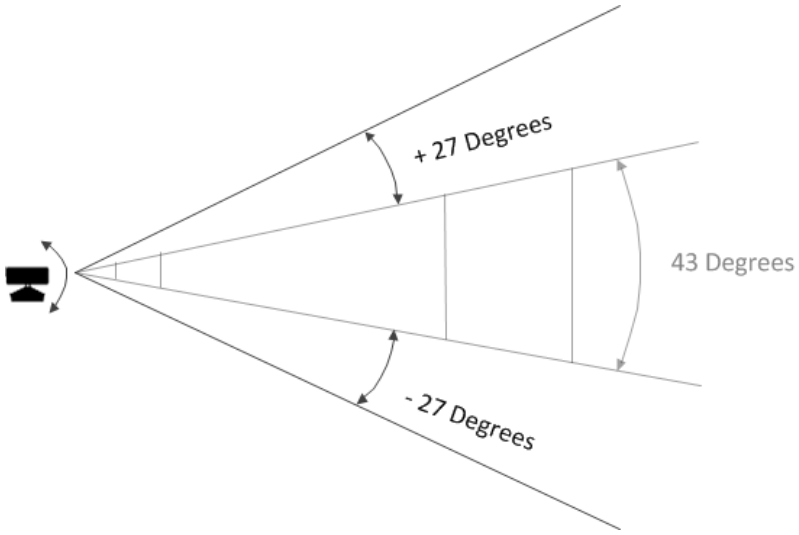
Zariadenie Kinect (Obrázok 1.2) vznikalo niekoľko rokov ako spolupráca oddelenia Microsoft Research (MSR) a izraelskej spoločnosti PrimeSense. Tá vytvorila referenčné zariadenie, ktoré obsahovalo RGB kameru, infračervenú kameru spolu so zdrojom infračerveného svetla. Toto zariadenie dokázalo spracovávať údaje hĺbky zo snímaného obrazu inovatívnym spôsobom. Proces snímania hĺbky z obrazu môžeme zjednodušene popísať ako vytvorenie mriežky (vzoru) z bodov infračerveného svetla. Meraním vzdialeností medzi týmito bodmi bolo možné vytvoriť mapu hĺbky vo veľkosti 320 x 240 pixelov. Zariadenie ďalej prepojí túto mapu spolu s obrazom z RGB kamery a infračervenej kamery (Obrázok 1.3) a získame tak RGBD dáta pre ďalšie spracovanie. Do produktu PrimeSense pridalo oddelenie MSR mikrofónové pole a vytvorilo tak smerový mikrofón, ktorý slúži na rozpoznávanie reči. Finálne zariadenie obsahuje aj motorček pre nastavenie náklonu zariadenia.


*Obrázok 1.2: Zariadenie Kinect (zdroj: [6])*



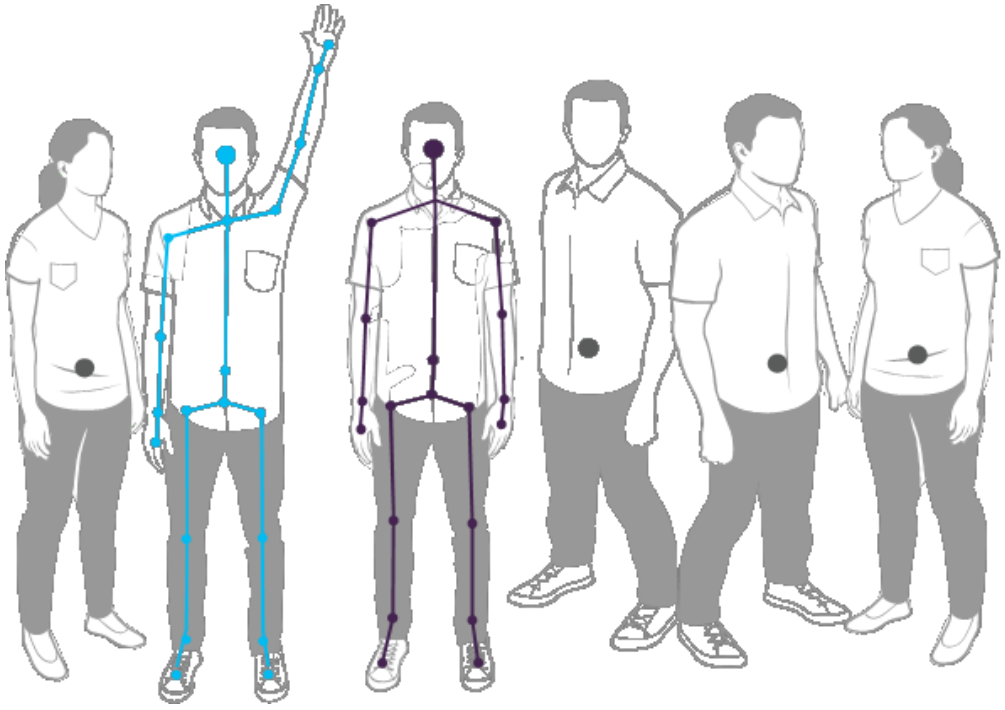

*Obrázok 1.3: Rozloženie hardvéru vo vnútri zariadenia Kinect (zdroj: [6])*

Pozorovací uhol zariadenia je 43° vertikálne a 57° horizontálne. Motorčekom pre nastavenie náklonu je možné zmeniť vertikálny pozorovací uhol o ± 27° (Obrázok 1.4). Snímacia frekvencia je 30fps, maximálne rozlíšenie pri tejto frekvencii 640 x 480px. Pri nižšej snímacej frekvencii je možné dosiahnuť rozlíšenie kamery až 1280 x 1024px, no na úkor presnosti detekcie pohybu osôb. Senzor získava dáta hĺbky obrazu pomocou monochromatického toku dát z infračervenej kamery v 11 bitoch, čo vytvára 2048 úrovní detekcie hĺbky obrazu. Rozsah detekcie osôb je vo vzdialenosti približne od 80 cm až po približne 400 cm. Mikrofónové pole je zložené zo štyroch mikrofónových kapsúl, kde každá z nich sníma oba kanály 16 bitového audia v snímacej frekvencii 16kHz. Viac informácií o špecifikácii senzora je možné získať tu [6].




*Obrázok 1.4: Pozorovacie uhly zariadenia Kinect (zdroj: [6])*

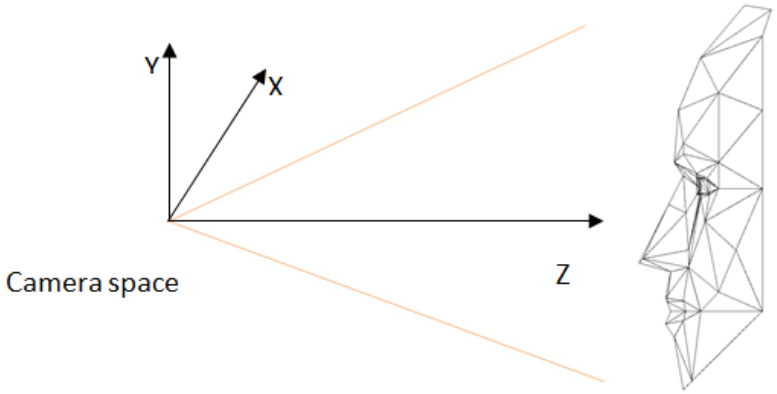
Kinect dokáže simultánne detegovať šesť osôb, no len dvaja z nich môžu byť aktívnymi (Obrázok 1.5). Znamená to, že Kinect dokáže súčasne analyzovať pohyb a gestá pre dve osoby, pre každého z nich deteguje a spracováva 20 definovaných bodov.



*Obrázok 1.5: Ukážka simultánnej detekcie osôb (zdroj: [6])*

Hardvérové zariadenie je podporované softvérovými riešeniami MSR na detekciu osôb, sledovanie pohybu osôb (gestá) a rozpoznávanie hlasových príkazov. Pre vytvorenie detekcie osôb sa na začiatku vývoja používala tzv. T póza osoby, upaženie (osoba stojaca so zdvihnutými rukami vo vodorovnej polohe) na prvotnú detekciu osoby stojacej pred senzorom. Toto gesto nebolo uspokojivé, preto sa vedci z MSR na čele z Jamie Shotton, rozhodli využiť nahrávky ľudí pri tancovaní, behaní či akrobacii, získaných prevažne z filmov a „naučiť" rozpoznávací softvér rozdeliť dáta prijaté zo senzora na použiteľné časti ľudského tela. Tak vytvorili mapovanie, ktoré dokáže rozdeliť prúd RGBD dát na 31 častí tela, bez nutnosti použitia pôvodnej T pózy, na prvotnú inicializáciu detekcie osoby. Viac o T póze sa čitateľ dozvie tu [2].

Kinect je možné využiť aj na rozpoznávanie tváre (Obrázok 1.6). Pomocou rozpoznávania tváre je možné určiť kedy má daná osoba otvorené oči alebo aký výraz tváre práve má (úsmev, zamračené, atď.). Vďaka analýze hĺbky obrazu je možné identifikovať konkrétnu tvár a vytvoriť tak jej 3D model. Prípadne je možné použiť tento model a mapovať ho na rôzne 3D modely vytvorené v grafickom programe.




*Obrázok 1.6: Rozpoznávanie tváre (zdroj: [6])*

Výstupom rozpoznávanie tváre pomocou zariadenia Kinect je:

- Stav rozpoznávania
- 2D body – 87 bodov, ktoré určujú ohraničenie tváre a jednotlivých častí ako oči, nos a ústa
- 3D pozícia hlavy – určuje zdvihnutie, natočenie alebo otočenie hlavy
- Jednotky animácie – určuje výraz tváre v rôznych druhoch

## 1.3 Použitie

Predstavením senzora spoločnosť Microsoft, priniesla nový spôsob interakcie s aplikáciami. Uvoľnením verzie Kinectu pre Windows pribudli aj nové možnosti použitia senzora v sférach priemyslu, zdravotníctva, vývoja, robotiky, atď.

### 1.3.1 Herná konzola

Primárne využitie pre Kinect je od začiatku vývoja v spojení s hernou konzolou Xbox. Priamo tak konkuruje iným pohybovým ovládačom ako napr. Wii Remote Plus pre hernú konzolu Wii či ovládačom od spoločnosti Playstation – Playstation Move alebo Playstation Camera.

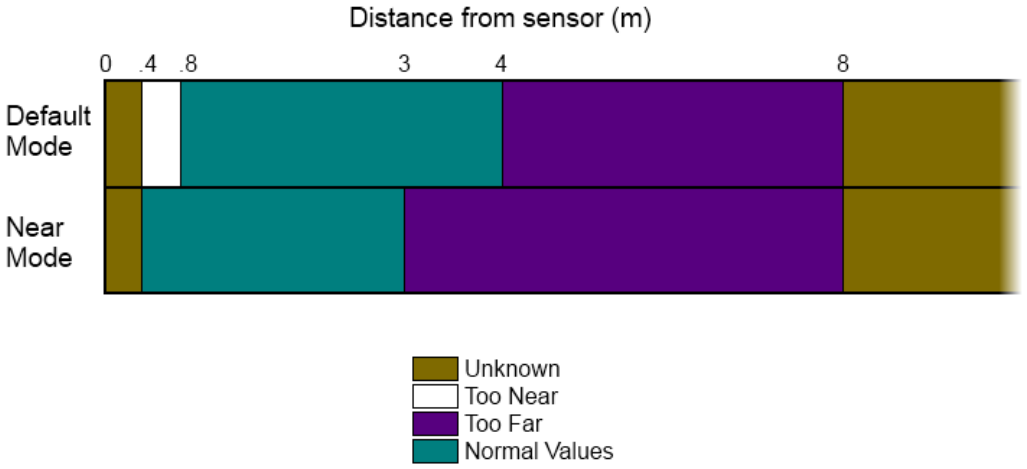
Medzi prvé hry vydané spoločne s uvedením zariadenia Kinect, boli hry ako Kinect Adventures a Kinect Sports, ktoré rôznym spôsobom využívali celé telo hráča na ovládanie samotnej hry.

### 1.3.2 Kinect pre Windows

31. októbra 2011 Microsoft predstavil komerčnú verziu zariadenia pod názvom „Kinect for Windows“ spoločne s uvoľnením potrebných vývojových nástrojov. Zariadenia Kinect pre hernú konzolu a Kinect pre Windows sú v oblasti hardvérovej výbavy úplne rovnaké.

Rozdiely medzi verziami je hlavne v použitej verzii firmvéru:

- Kinect pre Windows podporuje „blízky mód“, kedy senzor dokáže lepšie sledovať objekty už od 40cm. Avšak stráca tak schopnosť sledovať vzdialenejšie body. Rozloženie vzdialeností je ilustrované na Obrázok 1.7.




*Obrázok 1.7: Rozdiel sledovania objektov vo vzdialenosti od senzora (zdroj: [6])*

- „Seat mode“ – mód, ktorý poskytuje 10 bodov detekcie kostry a umožňuje tak sledovať osobu od pása nahor aj keď sedí.
- „Kinect Fusion“ – Kinect pre Windows dokáže okrem detekcie tváre mapovať aj iné objekty, či celý priestor do 3D, s ktorým je možné ďalej pracovať.
- Detekcia zovretia ruky – pomocou detekcie ruky je možné implementovať gestá NUI ako napr. „pinch-to-zoom“ (roztiahnutie rúk pre priblíženie obrazu), uchytenie a preloženie objektu a iné.
- Rozšírené možnosti nastavenia kamery – vďaka povoleným možnostiam v rámci firmvéru, je možné meniť jas, expozíciu a množstvo ďalších nastavení pre snímanie pomocou Kinectu pre Windows.

Využívanie rozšíreného firmvéru je možné vidieť najmä v komerčnej sfére. V prostrediach, kde je vhodné využiť bezdotykové ovládanie ako napr. operačné sály, kde si za použitia Kinectu môže lekár počas operácie prezrieť röntgenové snímky pacienta, či vďaka iným podporným technológiám sledovať priebeh operácie vo zväčšenej mierke (Obrázok 1.8).

*Obrázok 1.8: Ukážka použitia Kinect pre Windows v nemocniciach*[1]

Ďalším využitím môže byť výber sortimentu v nákupných centrách, hlavne v obchodoch s oblečením kde je možné prezrieť si tovar, bez nutnosti skúšania každého druhu sortimentu. Kinect pre Windows má veľké množstvo použití aj v akademickej (výuka angličtiny pre deti za pomoci rozpoznávania hlasových príkazov, pohybová interakcia spoločne s vývojom motoriky detí), priemyselnej (vývoj zariadenia na ovládanie strojov) či umeleckej sfére (maľovanie, tvorba 3D grafiky, tanec).

### 1.3.3 **Iné**

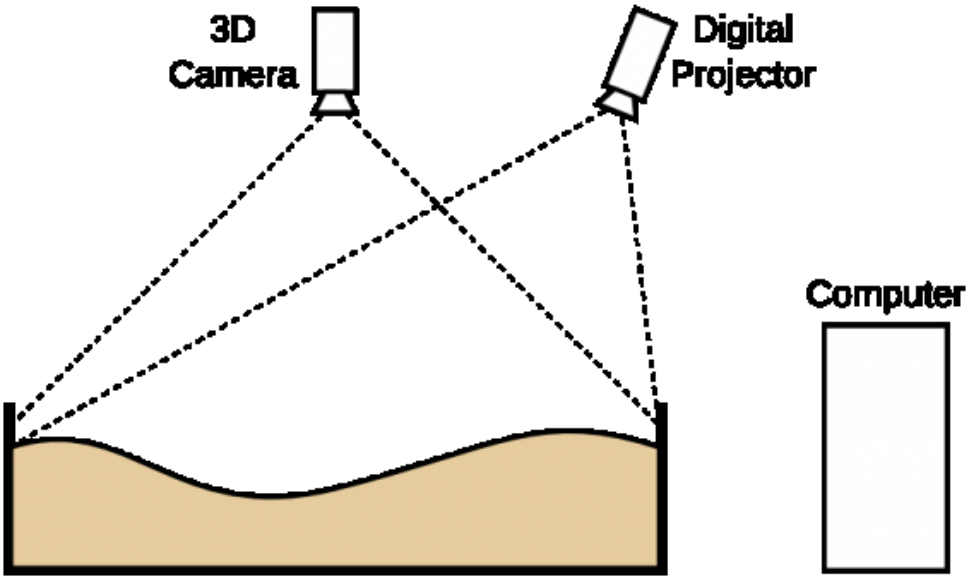
Okrem detekcie osôb a ich gest, ktoré ovládajú počítačové aplikácie je možné využiť Kinect aj iným spôsobom. Dôkazom je napr. ovládanie robotov, ktoré vďaka analýze hĺbky obrazu dokáže navigovať robota tak aby sa vyhýbal aj nečakaným prekážkam. Za pomoci senzora Kinect, môžeme robota ovládať aj hlasovými príkazmi (Obrázok 1.9).

*Obrázok 1.9: Ukážka robota, ktorý využíva zariadenie Kinect*[2]

Ďalším zaujímavým využitím zariadenia Kinect je spojenie 3D výstupu zo senzora, spoločne s projektorom. Takto je možné vytvoriť napríklad interaktívne pieskové ihrisko. Ukážka návrhu takého ihriska sa nachádza na Obrázok 1.10.

---

[1] Clean'Move, http://apps.after-mouse.com/fiche-produit/clean-move.html#
[2] RoboCar 1/10 Lite, http://www.zmp.co.jp/products/robocar-110lite




*Obrázok 1.10: Pieskové ihrisko s využitím zariadenia Kinect a projektora*[3]

[3] Pieskové ihrisko, http://itcolossal.com/sandystation/

# 2 Technológie pre tvorbu systému hudobného sprievodu

Výsledkom tejto diplomovej práce je systém, ktorým bude možné vytvoriť hudobný podklad za pomoci gest rozpoznávaných pomocou zariadenia Kinect. S využitím komunikácie cez internetové pripojenie má systém zabezpečiť vytváranie hudby pomocou viacerých zariadení Kinect.

Systém tak bude pozostávať z komunikačnej služby a k nej pripojených aplikácií na platforme OS Windows. Tieto aplikácie budú rozdelené do dvoch druhov. Aplikácie, ktoré zabezpečia komunikáciu so senzorom Kinect a aplikácie, ktoré budú vytvárať na základe prijatých dát hudobný sprievod.

Komunikačná architektúra práce bude podľa zadania implementovaná pomocou technológie WCF. Pre vytvorenie klientských aplikácií existuje niekoľko možností návrhu a použitia, ktoré si v tejto kapitole predstavíme.

## 2.1 WCF

WCF – Windows Communication Foundation [7] – je rozhranie na tvorbu SOA (z angl. service-oriented applications – aplikácie využívajúce služby) aplikácií, ktoré poskytuje jednoduchý prístup pre vytváranie webových služieb pomocou riadeného kódu. WCF veľmi uľahčuje vývoj koncových bodov komunikácie, predovšetkým preddefinovanými konfiguráciami a správou o zabezpečenie komunikácie.

Koncové body slúžia vo WCF na prepojenie klienta so službou. Koncový bod má adresu URL, ktorá špecifikuje, kde bude dostupný a vlastnosti väzby, ktoré špecifikujú ako budú dáta prenášané.

Použitím WCF je možné posielať dáta ako asynchrónne správy z jedného koncového bodu do druhého. Jedným z koncových tak môže byť služba, ktorá je spustená na servery a druhým môže byť klientska aplikácia spustená v počítači či mobilnom zariadení. Samotná klientska aplikácia môže pracovať s niekoľkými službami naraz a naopak služba môže poskytovať informácie niekoľkým klientom naraz.

Vlastnosti WCF:

- Interoperabilita – WCF implementuje moderný prístup pre prácu s webovými službami na základe štandardov.
- Podpora rôznych vzorov pre posielanie správ – správy, ktoré si vymieňa klient a server môžu byť podporované viacerými vzormi. Najčastejšie je to vzor požiadavka/odpoveď, ale aj jednosmerná správa či asynchrónne posielanie správ.
- Metadáta služby – WCF podporuje publikovanie metadát v rôznych štandardoch, ako napr. WSDL, XML Schéma a WS-politika.
- Dátové kontrakty – WCF je postavená na rozhraní .Net a teda obsahuje metódy na vytváranie kontraktov pomocou tried, ktoré reprezentujú dátové entity.
- Zabezpečenie – pre ochranu súkromia môžu byť správy kryptované a taktiež, je možné vyžadovať autentifikáciu od užívateľov.

- Rôzne spôsoby kódovania a prenosu správ – najbežnejší spôsob posielania správ je pomocou SOAP správ cez HTTP protokol. WCF však umožňuje napríklad kódovanie správ do formátu JSON a spôsob prenosu cez TCP, pomenované kanály, MSMQ či v novej verzii aj web socket.
- Rozšíriteľnosť a iné – WCF poskytuje množstvo iných vlastností a taktiež aj rozšíriteľnosť.
- Štandardizácia – WCF implementuje niekoľko rôznych WS (z angl. Web service – webová služba) štandardov, ktoré zabezpečujú samotnú komunikáciu či rozširujú jej vlastnosti, napr. WS-Addressing, WS-Security, ale aj RSS služby, smerovanie či podporu REST služieb.

### 2.1.1 Konfigurácia a spustenie

Umiestnenie služby WCF je možné ako proces na server IIS alebo ako samostatne bežiacej služby v operačnom systéme. Keďže WCF poskytuje veľké možnosti konfigurácie a prispôsobenia, je možné vytvoriť rôzne spojenia služby, klientov a ich vzájomnej komunikácie.

Prenos, kódovanie a detaily prenosového protokolu sa definujú pomocou konfigurácie väzby (z angl. binding). Binding reprezentuje vo WCF vrstvu, ktorá zabezpečuje prepojenie koncových bodov, teda je nutné aby jednotlivé časti podporovali daný typ väzby. Najjednoduchší spôsob pre zabezpečenie spojenia dosiahneme rovnakou konfiguráciou bindingu oboch strán komunikácie.

Konfigurácia bindingu a ostatných častí komunikácie je v projekte umiestnená:

- v klientskej aplikácií je to súbor App.config.
- WCF služba podporuje dva spôsoby konfiguračného súboru na základe prostredia v ktorom je spustená:
    - ak ide o IIS, tak používa Web.config
    - ak je použité iné prostredie tak App.config.

Oba konfiguračné súbory sú XML súbormi, ktoré majú podobnú štruktúru a obsahujú rôzne sekcie (Výpis kódu 2.1), ktoré definujú nastavenia nie len pre WCF.

```
<system.ServiceModel>
  <services/>
  <bindings/>
  <behaviors/>
</system.ServiceModel>
```

*Výpis kódu 2.1: Hlavné sekcie konfigurácie pre komunikáciu s WCF.*

WCF umožňuje použitie preddefinovaných spôsobov väzby a povoľuje aj vytvorenie vlastného spôsobu bindingu.

Samotný binding element obsahuje tri typy elementov:

- Binding element protokolu komunikačného kanálu – definuje zabezpečenie, spoľahlivosť, nastavenia toku kontextu či užívateľsky definované protokoly.

- Element prenosového kanálu – definícia základného transportného protokolu, ktorý zabezpečuje posielanie správ koncovému bodu, napríklad TCP alebo HTTP.
- Element na kódovanie správ – ktorý definuje, v akom formáte budú správy posielané, napr. XML, binárny formát alebo MTOM.

Najčastejšie využívané preddefinované typy konfigurácie väzby:

BasicHttpBinding – navrhnutý pre komunikáciu so službou, ktorá pracuje so základným profilom. Táto konfigurácia využíva HTTP protokol a textový alebo XML typ kódovania správ, napr. ASP.Net Web služba (ASMX).

WSHttpBinding, WSDualHttpBinding, - reprezentuje konfiguráciu, ktorá poskytuje zabezpečený a interoperabilný binding, ktorý podporuje distribuované transakcie. „Dual“ znamená, že daná konfigurácia podporuje duplexnú komunikáciu.

NetTcpBinding – zabezpečený binding optimalizovaný pre komunikáciu medzi WCF aplikáciami na rôznych zariadeniach.

NetNamedPipeBinding – binding, ktorý je navrhnutý na komunikáciu medzi WCF aplikáciami spustenými na jednom zariadení.

NetMsmqBinding – nastavenie podporuje MSMQ prenos, teda radenie komunikácie no radu.

NetPeerTcpBinding – podporuje infraštruktúru peer-to-peer na TCP vrstve.

WebHttpBinding – konfigurácia využíva na komunikáciu medzi koncovými bodmi požiadavky http protokolu namiesto SOAP správ.

NetHttpBinding –pribudla ku preddefinovaným konfiguráciám bindingu v .NET Frameworku vo verzii 4.5. Táto konfigurácia je navrhnutá na pre používanie s HTTP alebo websocket službami a štandardne používa binárne kódovanie. Využitím protokolu websocket je hlavne v komunikácii medzi webovou stránkou a serverom, často sa však používa aj v rôznych klientských aplikáciách. Protokol websocket umožňuje plne duplexnú komunikáciu cez jedno TCP spojenie. Tento binding sám rozhodne či použije kontrakt pre požiadavku-odpoveď alebo duplexné nastavenie. Toto nastavenie je však možné meniť.

Každá preddefinovaná konfigurácia poskytuje rôzne nastavenia pre prenos, zabezpečenie komunikácie, kódovanie správ a iné. Tieto konfigurácie je možné podľa potreby upraviť, či vytvoriť vlastné komunikačné nastavenie.

WCF poskytuje rôzne možnosti na vytvorenie inštancie služby:

- Per call – inštancia služby sa vytvára každým volaním metódy v službe.
- Per session – všetky volania sa vykonávajú v rámci jedného volania, no každý pripojený klient má vlastnú inštanciu.
- Single instance – jedna inštancia, ktorá sa vytvorí pre všetkých klientov a obstaráva tak všetky metódy, ktoré sú volané ako jediná.

Pre správnu komunikáciu je však nutné zabezpečiť aby služba dokázala spracovať viacej požiadaviek v rovnakom čase. Preto WCF umožňuje určiť tzv. súbežnosť (z angl. concurrency):

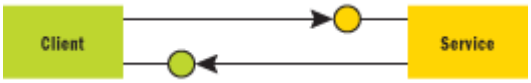
- Single – WCF spracováva iba jednu požiadavku v danom čase, ostatné požiadavky musia čakať, kým služba túto požiadavku nespracuje.
- Multiple – pre každú požiadavku si služba vytvorí nové vlákno. Takto môže spracovať niekoľko požiadaviek v rovnakom čase. Je však potrebné zabezpečiť problémy, ktoré vznikajú s viac vláknovým prístupom k objektom.
- Reentrant – všetky požiadavky obstaráva jedno vlákno, no služba sa postará o uvoľnenie vlákna a zámkov ak je v rámci služby volaná iná WCF služba.

Rôznym nastavením inštancií a spracovávania požiadaviek tak vznikne 9 možných kombinácií, ktoré môžeme použiť pre komunikáciu so službou. Toto nastavenie upresňuje, ako sa komunikácia medzi klientami vykonáva.

Klasický návrh aplikácií pracuje s jednoduchým spôsobom akým klient volá metódy, ktoré služba poskytuje. Klient vytvorí požiadavku na službu volaním metódy s vložením potrebných parametrov. Kým služba požiadavku spracuje, klient ostáva blokovaný (pokiaľ nejde o asynchrónne volanie, aj tu sa však stále očakáva výsledok operácie). Keď služba pošle odpoveď, klient s výsledkom ďalej pracuje.

WCF však podporuje ďalšie dva typy operácií, ktorými je možné so službou komunikovať:

- Jednosmerné operácie (z angl. one-way operations) – vytvárajú podporu pre operácie, ktoré je možné vyvolať, no klient sa už nestará o ich výsledok (z angl. fire-and-forget). Tieto operácie nenahrádzajú asynchrónne volania. Výsledok operácie nevráti žiadnu hodnotu a taktiež nevráti žiadne chybové hlásenie.
- Duplexné „callback“ operácie (z angl. callback – spätné volanie) – WCF dovoľuje službe volanie pripojených klientov. V prípade callbacku sa v podstate vymenia roly a klient sa stane službou a služba klientom (Obrázok 2.1).




*Obrázok 2.1: Callback vo WCF službe (zdroj: [8])*

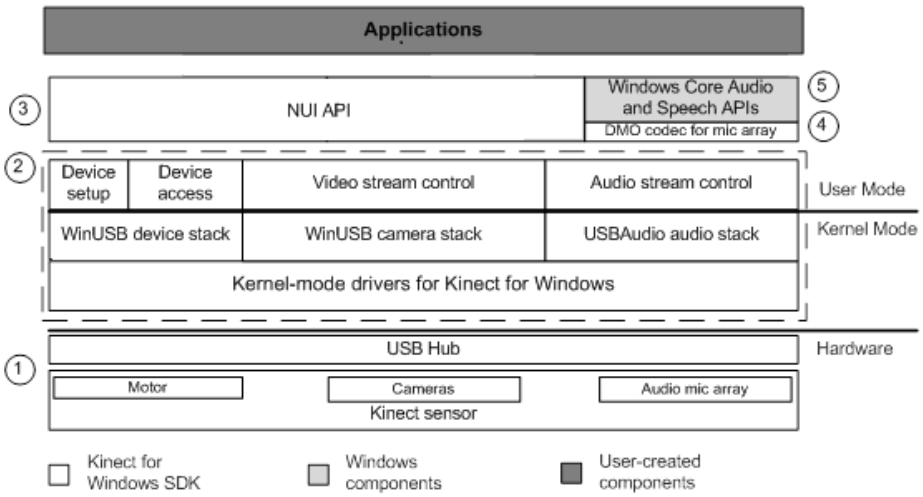
Callback operácie tak simulujú udalosti (z angl. event), na ktoré potom môžeme v klientskej aplikácii reagovať. Nie je tak nutné čakať na výsledok operácie. Volanie callbacku môžeme vykonať kedykoľvek v rámci spojenia klient-server. Pomocou callbacku je možné vyvolať udalosť na viacerých klientoch naraz. Teda môžeme implementovať prístup vydavateľ-odoberateľ (z angl. publish-subsribe), kde sa klienti pripojením k serveru stanú odoberateľmi dát ktoré server vystaví na callback kanál. Viac informácií o prístupe publish-subscribe je možné získať tu [8].

Publish-subscribe princíp sa používa vo veľkej miere pri mobilných klientoch akými sú napr. smartfóny. Hlavné využitie má pri notifikáciách, kedy sa klient prihlásením k službe poskytovateľa (zaregistrovaním na stránkach a podobne) prihlási k odberu notifikácií, ktoré

sú buď určené všetkým užívateľom alebo len priamo jemu. Server tak môže odlíšiť podľa implementácie, ako a kam má doručiť správy pomocou kanálov pre spätné volanie.

## 2.2 SDK pre Kinect

Pre natívny aj riadený vývoj poskytuje spoločnosť Microsoft vývojové knižnice, ktoré sú potrebné na vývoj aplikácií pre Kinect – SDK – Software development kit [6]. SDK predstavuje balík rôznych API (Obrázok 2.2) potrebných pre ovládanie samotného senzora ale aj množstvo funkčných častí, ktoré uľahčujú vývoj aplikácií vývojárom. SDK je dostupné na internetových stránkach spoločnosti Microsoft[4]. V rámci inštalácie SDK sú taktiež nainštalované aj ovládače pre komunikáciu so zariadením Kinect. SDK pre Kinect pre Windows, ktorý je kompatibilný so zariadením Kinect pre XBox 360 je vo verzii 1.8.




*Obrázok 2.2: Architektúra SDK (zdroj: [6])*

### 2.2.1 Systémové požiadavky

Vývoj pomocou SDK pre Kinect je možný na OS Windows od verzie Windows 7. Na vývoj je možné využiť Visual Studio 2010 a novšie; a .NET Framework 4 a novší.

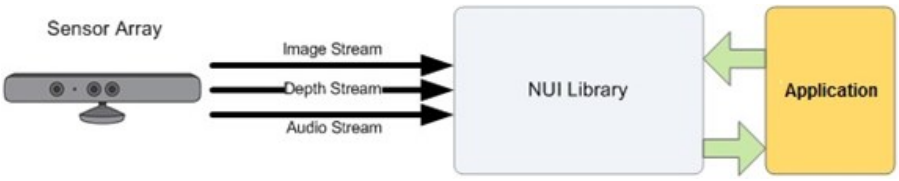
Minimálna hardvérová konfigurácia:

- 32/64 bitový procesor
- Dvojjadrový procesor s minimálnym taktom 2,66 GHz
- USB 2.0 pre pripojenie senzora
- Minimálne 2GB RAM pamäte
- Grafická karta s podporou DirectX 9

[4] Odkaz na stiahnutie SDK, https://www.microsoft.com/en-us/download/details.aspx?id=40278

### 2.2.2 Vývoj pomocou SDK

Samotné SDK podporuje vývoj v natívnom jazyku C++ a riadených jazykoch C# a Visual Basic. Pre vývoj v jazyku C# je potrebné pridať referenciu na knižnicu Microsoft.Kinect.dll.




*Obrázok 2.3: Interakcia hardvéru a softvéru (zdroj: [6])*

Senzor Kinect poskytuje pre SDK vstupné prúdy dát. Ovládače sa starajú o prístup k mikrofónom, ovládanie audio a video dátových prúdov či funkcie pre správu viacerých pripojených zariadení (Obrázok 2.3). SDK obsahuje objekty pre média DirectX (DMO – z angl. DirectX Media Objects) pre lokalizáciu zdroja zvuku za pomoci mikrofónového poľa. V SDK sú taktiež aj štandardné API pre podporu spolupráce s OS.

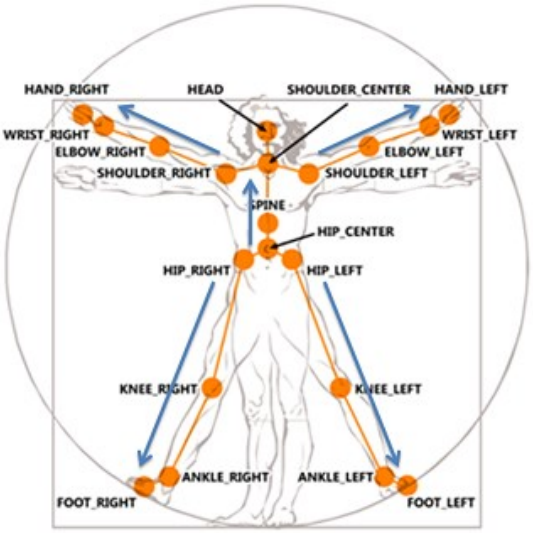
SDK poskytuje možnosť programovo kontrolovať a pristupovať k dátovým prúdom, ktoré senzor poskytuje:

- Audio – na implementáciu a použitie zvukových dát je v riadenom kóde možné využiť DMO.
- Color (farebný, video prúd) – Kinect poskytuje video prúd v rôznych rozlíšeniach a formátoch (RGB, YUV, Bayer). SDK poskytuje možnosť na určenie typu, ktorým je možné vybrať druh kódovania dátového prúdu.
  Špecifikácia formátu pre riadený kód:
  - RgbResolution640x480Fps30
  - RgbResolution1280x960Fps12
  - RawYuvResolution640x480Fps15
  - RawBayerResolution1280x960Fps12
  - RawBayerResolution640x480Fps30

  Výber rozlíšenia a rýchlosti snímkovania ovplyvňuje množstvo dát, ktoré je nutné preniesť zo senzora do aplikácie cez USB rozhranie. Dátový tok rozhrania je však obmedzený, čo ovplyvňuje rýchlosť obrazu v aplikácii. Znížením rozlíšenia, zmenšíme množstvo dát prenášaných cez USB, vďaka čomu je obraz plynulejší.
- Depth (hĺbka) – každá snímka dátového prúdu obsahuje dáta o hĺbke obrazu, kde každý pixel obrazu obsahuje informáciu o vzdialenosti od kamery k najbližšiemu snímanému objektu v milimetroch. Prúd dát okrem hĺbky obsahuje aj dáta o segmentácii osoby stojacej pred senzorom. Tieto dáta môžeme použiť na izoláciu špecifického užívateľa alebo oddelenie regiónov.
- Infrared (infračervený) – Senzor generuje infračervené svetlo, za pomoci ktorého určuje vzdialenosť objektu od senzora. Primárne použitie infračerveného dátového prúdu slúži na

kalibráciu kamery použitím testovacieho vzoru získaného z farebného a infračerveného prúdu. Infračervený prúd nie je skutočným oddeleným dátovým prúdom, no čiastočne pozmenenými dátami video prúdu.

- Skeleton (kostra) – dátový prúd s informáciami o pozícii a postoji osôb, stojacích pred zariadením. Senzor dokáže vďaka infračervenému svetlu sledovať až 6 osôb, no len pre dve osoby poskytuje informácie o postoji a polohe rúk, nôh a hlavy. Aplikácia tak môže lokalizovať jednotlivé body ľudského tela a sledovať tak ich pohyb (Obrázok 2.4).




*Obrázok 2.4: Pozície jednotlivých bodov ľudského tela, ktoré senzor sleduje (zdroj: [6])*

Ovládače ktoré sú súčasťou SDK dovoľujú komunikáciu s viacerými senzormi naraz. API obsahuje funkcie, ktorými môžeme špecifikovať, ktoré zariadenia bude naša aplikácia používať. Pre každý jeden senzor môžeme taktiež nastaviť rôzne parametre.

Trieda KinectSensor obsahuje kolekciu všetkých dostupných senzorov (Výpis kódu 2.2), ktoré SDK rozpozná.

```
private KinectSensor sensor;
foreach (var potentialSensor in KinectSensor.KinectSensors)
{
        if (potentialSensor.Status == KinectStatus.Connected)
        {
                this.sensor = potentialSensor;
                break;
        }
}
```

*Výpis kódu 2.2: Výber potencionálneho senzora*

Po výbere vhodného senzora (môže ich byť aj viac), je ďalším krokom spustenie a povolenie rôznych typov prúdov (Výpis kódu 2.3), ktoré bude senzor snímať a prenášať do aplikácie.

```
if (this.sensor != null)
{
        this.sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
        this.sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
        this.sensor.SkeletonStream.Enable();
        this.sensor.ColorStream.Enable(ColorImageFormat.InfraredResolution640x480Fps30);

        this.sensor.AudioSource.Start();
}
```

*Výpis kódu 2.3: Výber a inicializácia dátových prúdov, ktoré bude aplikácia používať*

V prípade dátových prúdov `ColorStream` a `DepthStream` určujeme v metóde `Enable` vstupný parameter s typom formátu, ktorý budeme chcieť využívať. Pre dátový tok zvuku sa používa na inicializáciu metóda `Start`.

Po nastavení a inicializácií dátových prúdov je potrebné spustiť snímanie senzora (Výpis kódu 2.4).

```
if (this.sensor != null)
{
        this.sensor.Start();
}
```

*Výpis kódu 2.4: Spustenie snímania senzora*

Na využitie rôznych dátových prúdov poskytuje trieda `KinectSensor` konkrétne vlastnosti a udalosti. Ich využitím tak môžeme pristupovať ku konkrétnym dátam, ktoré senzor poskytuje. Na prístup k jednotlivým snímkam využijeme udalosti, na ktoré je možné zaregistrovať našu metódu.

Udalosť `ColorFrameReady` sa vyvolá keď je dostupná nová snímka v dátovom prúde pre video. Rovnako tak aj pre iné dátové prúdy ako `DepthFrameReady` a `SkeletonFrameReady`. Je však možné použiť aj udalosť `AllFramesReady`, ktorá sa vyvolá vždy keď sú dostupné dáta pre každý aktívny dátový prúd (Výpis kódu 2.5). V metódach, ktoré takto vyvoláme, danú snímku už môžeme spracovať a vyhodnotiť. Pre dátový tok zvuku poskytuje `KinectSensor` vlastnosť `AudioStream`, kde pomocou metódy `Read`, môžeme získať zvukové dáta zo senzora (Výpis kódu 2.5).

```
if (this.sensor != null)
{
        this.sensor.ColorFrameReady += this.SensorColorFrameReady;
        this.sensor.DepthFrameReady += this.SensorDepthFrameReady;
        this.sensor.SkeletonFrameReady += this.SkeletonFrameReady;
```

```
        int readCount = this.sensor.AudioStream.Read(audioBuffer, 0, audioBuffer.Length);
}
```

*Výpis kódu 2.5: Registrácia na udalosti dátových prúdov.*

Spracovanie a použitie získaných dát z dátových prúdov je už predmetom účelu aplikácie (Výpis kódu 2.6).

```
private void SensorColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
        using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
        {
                if (colorFrame != null)
                {
                        colorFrame.CopyPixelDataTo(this.colorPixels);
                }
        }
}
```

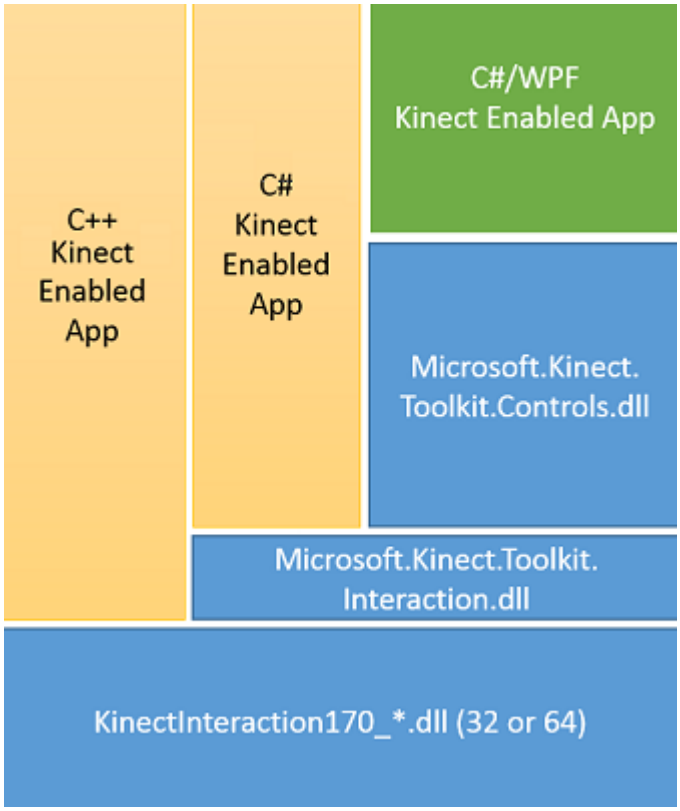
*Výpis kódu 2.6: Ukážka použitia dátového toku pre video a spracovania získaných dát*

### 2.2.3 Ďalšie možnosti vývoja pomocou SDK

Okrem hlavných funkcií na správu a ovládanie dátových prúdov (NUI) a senzora poskytuje SDK množstvo ďalších API a nástrojov na uľahčenie a skvalitnenie vývoja aplikácií.

Ovládanie motora na zmenu náklonu senzora je jednou z funkcií, ktorú senzor poskytuje. Vďaka vlastnosti `ElevationAngle` je možné definovať uhol náklonu senzora v aplikácií v rozmedzí -27° až 27°.

Nástroje KinectInteraction (Obrázok 2.5) nie sú priamou súčasťou knižníc SDK (tvoria ich samostatné knižnice), no poskytujú rozšírené možnosti detekcie gest a sledovanie pohybu a stavu dlane. Použitie API zabezpečíme pridaním referencie na .dll súbory.




*Obrázok 2.5: Architektúra KinectInteraction (zdroj: [6])*

Vďaka týmto nástrojom je možné sledovať dlaň – jej pozíciu, stlačenie (posun dlane smerom k senzoru v danom bode), posúvanie, uchytenie a pustenie. Táto funkcia je v spojení s ovládacími prvkami vhodná na ovládanie užívateľského rozhrania aplikácie. Využitie tejto interakcie v aplikácii je možné použitím toku dát z `InteractionStream`. Použitie tohto dátového prúdu je podobné ako pri dátových prúdoch poskytovaných SDK.

## 2.3 OpenKinect[5]

Okrem oficiálnej podpory od spoločnosti Microsoft a jej SDK existuje komunita ľudí, ktorá vytvorila otvorené zdrojové kódy a knižnice na využitie senzora Kinect. Vďaka týmto knižniciam je možné využiť zariadenia aj na iných operačných systémoch a vývoj aplikácií je možný pomocou ďalších rozšírených programovacích jazykov.

API OpenKinect obsahuje rozšírenia pre jazyky:

- C/C++
- C#/VB.Net
- Java
- Python

[5] OpenKinect, http://openkinect.org/wiki/Main_Page

- JavaScript

Komunita experimentuje aj s možnosťami využitia API pre:

- OpenCV
- Matlab
- LabView
- A iné

Projekt začal v Novembri roku 2010, kedy vznikli prvé zdrojové kódy na prístup k dátam senzora spoločnosti PrimeSense. Po vzniku zariadenia Kinect, komunita zverejnila API, ktoré obsahovalo nástroje na detekciu kostry a iné nástroje. Komunita neprestala pracovať na otvorených zdrojových kódoch ani po vydaní oficiálneho SDK. Najmä vďaka tomu vznikla podpora senzora pre rôzne platformy a vývoje jazyky.

## 2.4 Klientská aplikácia

Vytvorenie klientských aplikácií na platforme .Net je možné niekoľkými spôsobmi. Microsoft vo svojom OS Windows podporuje rôzne druhy programovacích jazykov a platforiem. Samozrejmosťou je podpora pre natívne aplikácie (C/C++), riadené kódy (C#, Visual Basic, Java, atď.) ale aj skriptovacie jazyky (JavaScript, Python a iné). Vo vývojom nástroji od spoločnosti Microsoft, Visual Studio, je taktiež podpora pre rôzne projekty na vytvorenie klientskej aplikácie (konzolové aplikácie, MFC, WinForms, WPF, Webové aplikácie a iné).

### 2.4.1 Konzolová aplikácia

Počítačový program, ktorý je navrhnutý na zobrazovanie informácií iba pomocou textu. Užívateľ komunikuje s programom pomocou textových príkazov zadávaných na klávesnici. V dnešnej dobe sú tieto aplikácie určené prevažne pre odborníkov a ako podpora grafického rozhrania aplikácie alebo ako testovacie aplikácie pre rôzne scenáre. Predstavujú tak rýchly spôsob akým zobraziť či upraviť požadované dáta bez načítavania knižníc potrebných na vykreslenie grafického rozhrania.

### 2.4.2 Windows Forms (WinForms)

Súčasťou :NET Framework rozhrania sú knižnice na vytvorenie GUI, ktoré sa nazývajú WinForms. Slúžia na vytvorenie klientských aplikácií v natívnom kóde (C# a Visual Basic) pre OS Windows. WinForms ponúka možnosť využitia rôznych ovládacích prvkov pre zobrazenie dát. Pre interakciu s rozhraním je možné používať reakcie na udalosti v rámci okna aplikácie.

Vývoj týchto aplikácií je v nástroji Visual Studio značne zjednodušený pomocou rôznych preddefinovaných ovládacích prvkov a ich jednoduchým implementovaním a použitím v aplikácii.

Všetky ovládacie prvky použité v aplikácii dedia z triedy Control, ktorá poskytuje minimálnu funkčnosť, ako napr. pozíciu, veľkosť, farbu a taktiež najčastejšie používané udalosti ako napr. kliknutie.

### 2.4.3 WPF – Windows Presentation Foundation

V .NET Frameworku vo verzii 3.0 Microsoft pripojil aj druhé API pre vývoj klientských aplikácií s GUI. WPF je grafický systém na tvorbu užívateľských rozhraní, ktorý na vykreslenie prvkov využíva DirectX. Na rozdiel od WinForms sa na definíciu vzhľadu a funkčnosti prvkov používa okrem riadeného kódu aj XAML (z angl. Extensible Application Markup Language), jazyk založený na XML.

WPF spája rôzne druhy elementov do jednotného systému na tvorbu rozhraní pre rôzne platformy. Užívateľské rozhrania tak môžeme vytvárať pre aplikácie Silverlight, aplikácie pre OS Windows, či mobilné aplikácie.

WPF podporuje návrh a dizajn pre rôzne rozloženie a vzhľad aplikácií, vďaka súborom šablón no aj s použitím vlastných komponent.

#### 2.4.3.1 *Programovanie WPF aplikácií*

XAML zabezpečuje lepšie oddelenie prezentačnej časti (Výpis kódu 2.7) od riadeného kódu, ktorý zabezpečuje spracovanie dát a logiku aplikácie (Výpis kódu 2.8). Je tak možné použitie rôznych dizajnových nástrojov a taktiež prenechať dizajn aplikácie grafikovi, zatiaľ čo programátor bude vytvárať logiku aplikácie. WPF prinieslo aj zjednodušenie globalizácie a lokalizácie aplikácií.

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        x:Class="Sample.AWindow"
        Title="Window with Button"
        Width="250" Height="100">

        <!-- Add button to window -->
        <Button Name="button" Click="button_Click">Click Me!</Button>

</Window>
```

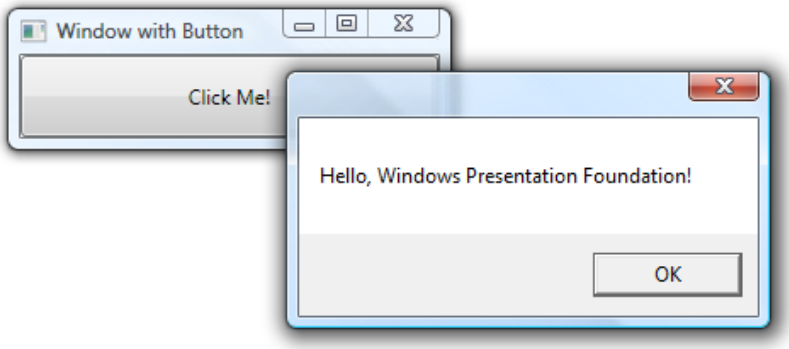
*Výpis kódu 2.7: Ukážka jednoduchej aplikácie v XAML*

```
namespace Sample
{
        public partial class AWindow : Window
        {
                public AWindow()
                {
                        InitializeComponent();
                }
                void button_Click(object sender, RoutedEventArgs e)
                {
                        MessageBox.Show("Hello, Windows Presentation Foundation!");
```

```
        }
    }
}
```

*Výpis kódu 2.8: Codebehind – časť WPF aplikácie s riadeným kódom*

Výsledkom je jednoduchá aplikácia (Obrázok 2.6) s jedným tlačidlom, ktoré po kliknutí vykoná naimplementovanú akciu.



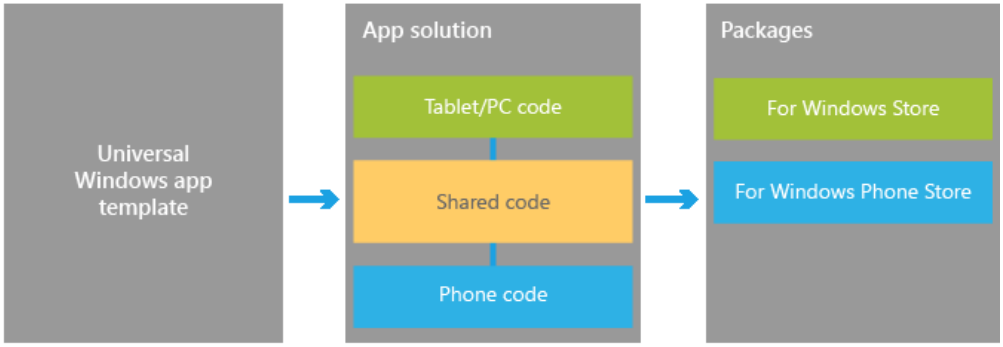

*Obrázok 2.6: Ukážka WPF aplikácie*[6]

### 2.4.4 Univerzálne aplikácie

Aplikácie, ktoré sa používajú v mobilných zariadeniach s OS Windows Phone a moderné aplikácie v systéme Windows 8.1 a vyššom (Obrázok 2.7). Ide o nový druh aplikácií, ktoré sú vyvíjané za pomoci WPF a kompilovaného kódu alebo HTML a JavaScriptu. No vzhľadom a ovládaním sa líšia od štandardných aplikácií. Sú prioritne určené pre dotykové ovládanie a ich dizajn by mal byť tomu prispôsobený. Je možné písať aj univerzálne aplikácie, ktorých riadený kód (príp. JavaScript) je jednotný pre obidve platformy, no prezentačná časť sa pre dané platformy líši.

Pre vývoj týchto aplikácií je potrebné Visual Studio 2013. Pre dizajnovanie aplikácií Microsoft poskytuje aj samostatnú aplikáciu s názvom Blend for Visual Studio, v ktorom je možné vytvárať prezentačnú stránku aplikácie. S vývojom týchto aplikácií Microsoft počíta do budúcnosti. Vytvárajú tak ekosystém aplikácií pre rôzne platformy.

[6] MSDN, WPF, https://msdn.microsoft.com/en-us/library/aa970268(v=vs.110).aspx




*Obrázok 2.7: Vývoj univerzálnych aplikácií*[7]

## 2.5 Audio v riadenom kóde

Systémová knižnica .NET Frameworku `System.Media`, obsahuje triedu `SoundPlayer`, ktorá slúži na základné operácie s podporovanými audio súbormi.

Pre rozšírenú prácu so súbormi existuje veľké množstvo voľne dostupných knižníc ale aj platených API. Každé z API poskytuje rozličné možnosti, funkcie a podporu práce so zvukovými súbormi. Najrozšírenejšími voľne dostupnými sú CSCore[8] a NAudio[9].

### 2.5.1 CSCore

Voľne dostupná .NET audio knižnica, ktorá je napísaná v jazyku C#. Je optimalizovaná pre výkon aplikácií a jednoduchá pre použitie v kóde. Podporuje veľa rôznych formátov zvukových súborov, výstupných a vstupných dátových vstupov či rôzne technológie tretích strán.

Použitie knižnice v programe je možné pomocou NuGet balíčka, ktorý CSCore poskytuje.

### 2.5.2 NAudio

NAudio je .NET audio a MIDI knižnica, ktorá obsahuje rôzne nástroje na prácu s audiom. Jej vývoj bol začatý v roku 2002. Poskytuje rôzne API na podporu vstupných a výstupných dát a ich spracovanie. Dokáže dekódovať a vytvárať rôzne hudobné súbory a podporuje rôzne audio formáty. Oproti CSCore je rozšírená o podporu a prácu s MIDI súbormi, obsahuje niekoľko WinForms ovládacích prvkov a dokáže mixovať niekoľko vstupných súborov do jedného výstupného dátového prúdu. Taktiež je možné využiť NuGet balíček pre integráciu NAudio knižníc do projektu.

---

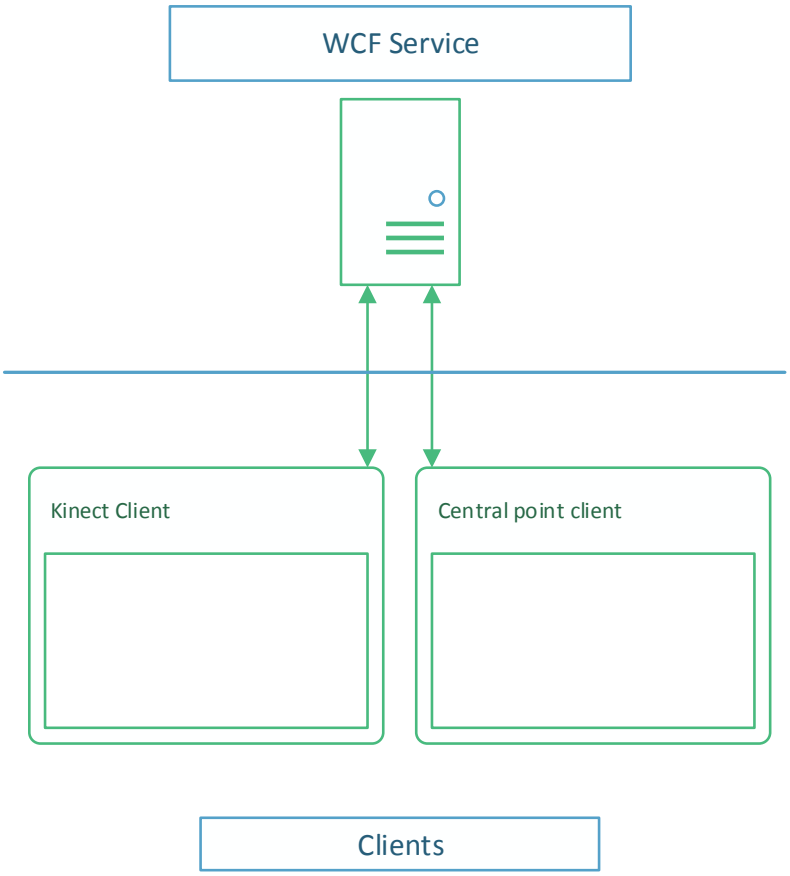
[7] Universal Windows App, http://blogs.msdn.com/b/cdndevs/archive/2015/02/23/how-to-choose-the-right-programming-languages-for-my-universal-windows-app-project.aspx

[8] CSCore, https://cscore.codeplex.com/

[9] NAudio, https://naudio.codeplex.com/

# 3 Návrh a realizácia systému hudobného sprievodu

Komunikačná architektúra systému pre hudobný sprievod pozostáva z troch oddelených blokov. Klient pre zariadenie Kinect, klient pre tvorbu hudobného sprievodu (centrálny bod) a WCF služby (Obrázok 3.1).




*Obrázok 3.1: Návrh systému*

Jednotlivé klientské aplikácie medzi sebou komunikujú cez WCF službu, ktorá je spustená na IIS servery. Pre potreby tejto práce som IIS server nainštaloval na Windows Server 2013, ktorý je umiestnený v službe Microsoft Azure[10], no pre testovanie aplikácie som často používal aj IIS server nainštalovaný na vlastnom zariadení s OS Windows 8.1.

Pre vytvorenie projektu som použil vývojové prostredie Visual Studio 2013. Celý projekt je napísaný v jazyku C# a za pomoci .NET Frameworku.

[10] Microsoft Azure, http://azure.microsoft.com/

## 3.1 WCF služba

V tejto práci použijem princíp publish-subscribe a teda využijem callback operácie na zabezpečenie interakcie centrálneho bodu a pripojených klientov a ich vzájomnej komunikácie.

V rámci služby WCF si budem ukladať informácie o callbacku pre pripojených klientov a centrálnych bodoch. Za pomoci týchto informácií budem môcť informovať jednotlivé spojenia o zmenách, ktoré vykonajú klienti či samotná WCF služba. WCF služba tak slúži aj na uchovanie dočasných informácií o pripojených klientoch.

Vďaka duplexnej podstate návrhu a použitia prístupu publish-subscribe, bolo nutné zvoliť správne nastavenia pre konfiguráciu služby. Duplexnú komunikáciu podporuje niekoľko štandardných väzieb. Zvolil som teda NetTcpBinding ako štandardný spôsob komunikácie a na overenie nových možností som využil aj NetHttpBinding. Tieto nastavenia použijem pri testovaní aplikácie a komunikačnej odozvy. Služba má v konfiguračnom súbore taktiež nastavenie pre zapnutie logovania chybových hlásení kvôli prípadným výnimkám v behu aplikácie.

V diplomovej práci som zvolil nastavenie služby (Výpis kódu 3.1), ktorá bude fungovať ako jedna inštancia po celú dobu svojej existencie (podľa nastavaní aplikačného prostredia IIS). A keďže bude nutné obstarávať niekoľko klientov, zvolil som viacnásobnú súbežnosť (z angl. Multiple concurrency).

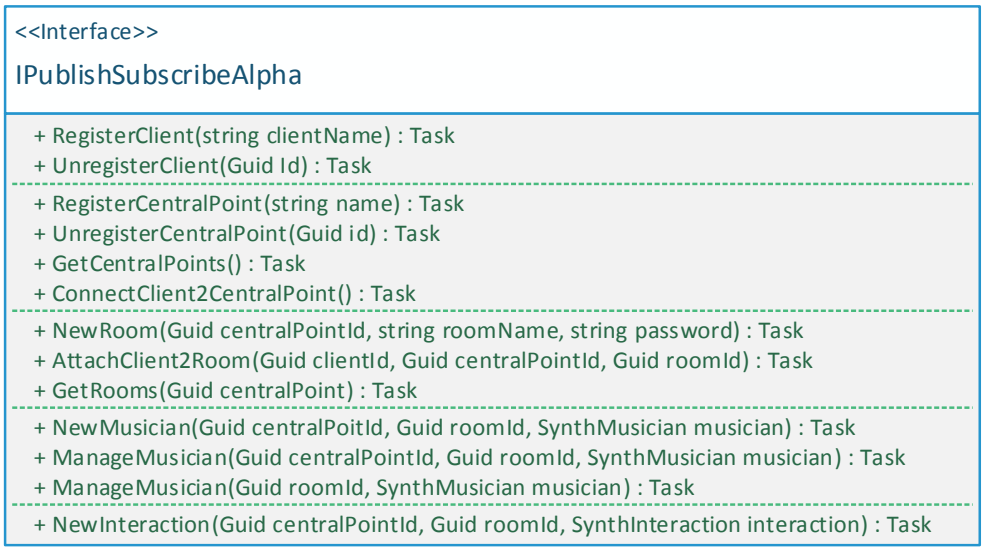
```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode = ConcurrencyMode.Multiple)]
```

*Výpis kódu 3.1: Atribúty služby*

Takto budú môcť jednotlivé časti komunikovať bez väčšieho spomalenia a čakania na uvoľnenie služby.

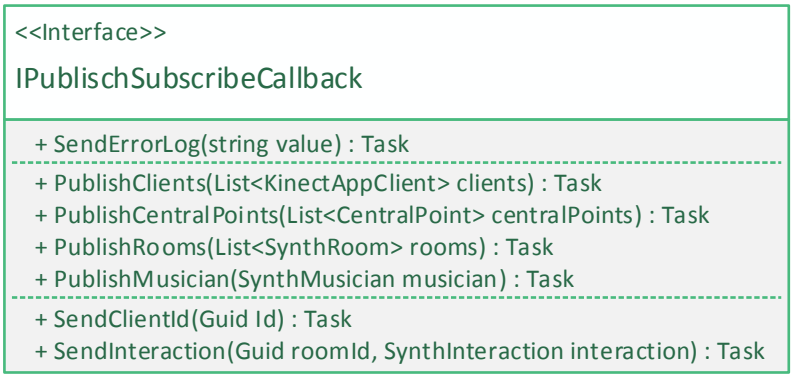
Samotná WCF služba obsahuje niekoľko metód (Obrázok 3.2), ktoré budú jednotlivý klienti využívať. Takto zabezpečím splnenie niekoľko kľúčových bodov:

- Pripojenie klientov k WCF službe.
- Pripojenie klienta k centrálnemu bodu.
- Vytvorenie identifikátorov pre oboch klientov a ich výmenu.
- Priradenie klienta s Kinectom k miestnosti.
- Priradenie osoby stojacej pred senzorom k centrálnemu bodu.
- Posielanie správ s informáciou o interakcii s Kinectom.




*Obrázok 3.2: Rozhranie WCF služby*

Ako je vidieť na obrázku rozhrania (Obrázok 3.3), sú všetky metódy služby implementované s návratovým typom `Task`. Trieda `Task` reprezentuje v .NET Frameworku asynchrónnu operáciu a predstavuje hlavnú komponentu asynchrónneho návrhového vzoru TAP (Task-based Asynchronous Pattern) [9]. Jednotlivé metódy tak nevracajú skutočnú odpoveď, to zabezpečíme v implementácii volaním metódy spätného volania. V publish-subscribe prístupe je teda nutné aby klientská aplikácia implementovala rozhranie pre callback.




*Obrázok 3.3: Rozhranie pre implementáciu spätného volania*

Získanie callback kanálu v službe WCF urobím pomocou prístupu k aktuálnemu prostrediu spustenia služby. Využijem na to systémovú triedu `OperationContext` (Výpis kódu 3.2).

```
OperationContext.Current.GetCallbackChannel<IPublishSubscribeCallback>();
```

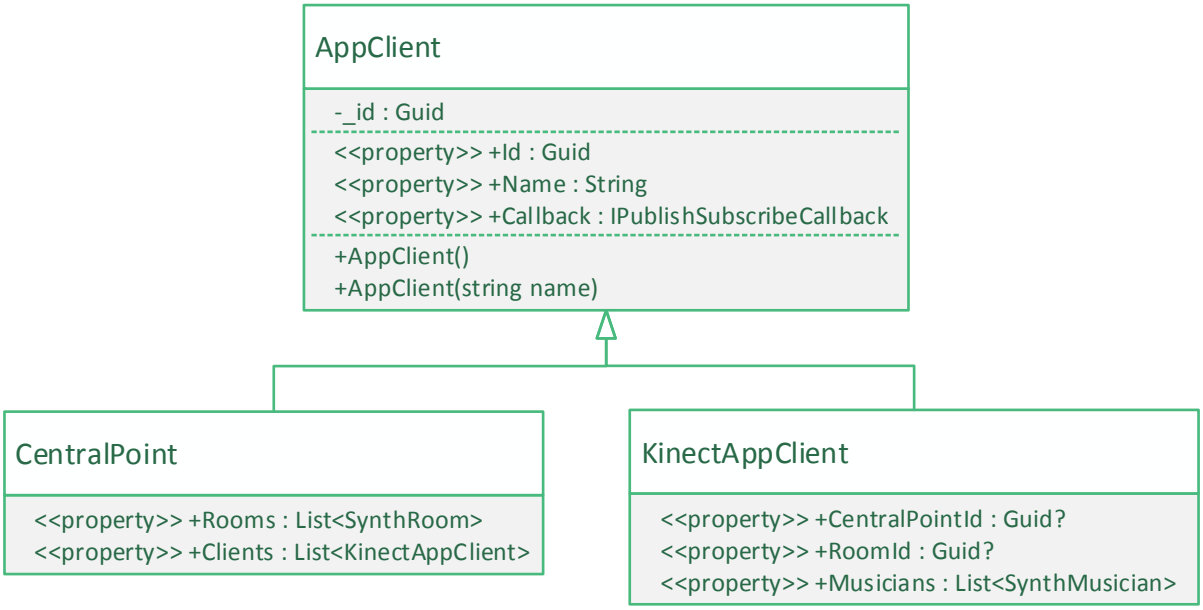
*Výpis kódu 3.2: Získanie callback kanálu*

V rámci WCF služieb je vytvorených niekoľko entít, ktoré vytvárajú logickú štruktúru a objekty používane v službe a klientských aplikáciách.

Zoznam entít používaných vo WCF službe:

- `AppClient`, `KinectAppClient`, `CentralPoint` – triedy pre identifikáciu a prácu s klientskými aplikáciami.
- `SynthRoom` – v rámci jednej inštancie centrálneho bodu je možné vytvoriť niekoľko miestností, ktorým môžeme nastaviť rôzne parametre a vlastnosti. K týmto miestnostiam sa potom jednotlivé klientské aplikácie pripájajú.
- `SynthMusician` – trieda osoby stojacej pred senzorom. Je použitá na presun informácií z klienta do centrálneho bodu a naopak, ktoré slúžia na identifikáciu osoby a na nastavenie parametrov.
- `SynthInteraction` – trieda, ktorá sa používa na prenos informácií o interakcií užívateľa s Kinectom.
- `SoundInfo` – trieda, ktorá nesie informácie o zvuku, ktorý sa má vykonať na základe akcie.

Triedy `KinectAppClient` a `CentralPoint` dedia z triedy `AppClient` (Obrázok 3.4). Tá obsahuje základné vlastnosti, ktoré sa používajú pre oba typy klientských aplikácií.




*Obrázok 3.4: Dedičnosť tried pre identifikáciu klienta v službe WCF*

V službe sú jednotlivé informácie na spätné volanie pre klientov ukladané do oddelených generických zoznamov:

- `List<KinectAppClient> _clients`
- `List<CentralPoint> _centralPoints`

Po pripojení klienta k WCF je prvým volaním (Obrázok 3.5) služby pre klientskú aplikáciu metóda `RegisterClient` s parametrom názvu. Táto metóda zaregistruje spätné volanie pre klientskú

aplikáciu a vyvolá callback so zaslaním identifikátora pre klienta. V prípade chyby s registráciou vyvolá callback pre s informáciou o chybe a chybu uloží do logu.

Pri ukončovaní klientskej aplikácie sa vždy vyvolá metóda na od registrovanie klientskej aplikácie `UnregisterClient` a WCF o tom informuje centrálny bod, ku ktorému bol klient pripojený.

Po zaregistrovaní klientov do WCF služby je potrebné ešte zabezpečiť priradenie klientskej aplikácie k centrálnemu bodu. Výsledkom úspešnej registrácie je identifikátor typu `Guid`. Tieto identifikátory využívam ako parametre v rôznych metódach. Taktiež aj na priradenie klientov v metóde `ConnectClient2CentralPoint`. Metóda má dva vstupné parametre a to identifikátor klientskej aplikácie a identifikátor centrálneho bodu. Pre získanie zoznamu dostupných centrálnych bodov, ku ktorým sa môže klient pripojiť sa využíva metóda `GetCentralPoints`.

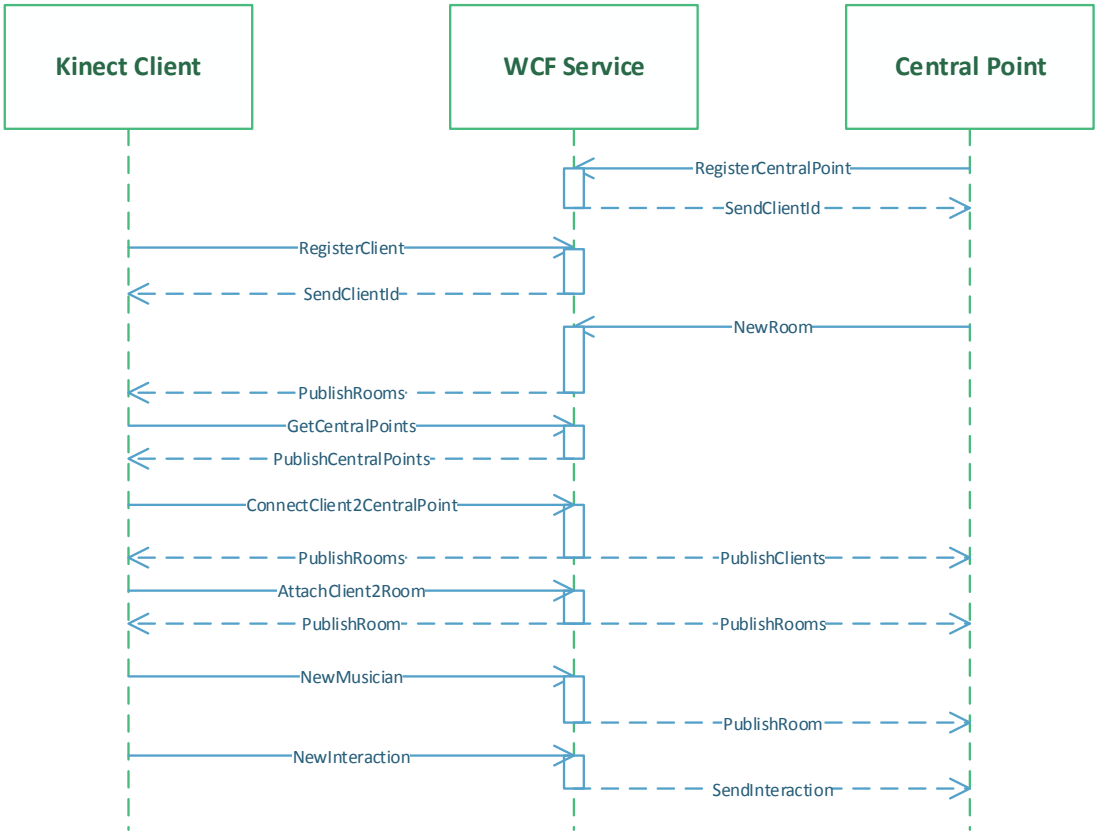
V rámci metódy priradenia klienta k centrálnemu bodu vyvolám použitím callback operácií:

- Aktualizáciu zoznamu pripojených klientov v aplikácii centrálneho bodu
- Pošlem klientskej aplikácii zoznam miestností, ktoré centrálny bod poskytuje

Následne je podobným spôsobom nutné priradiť klientskú aplikáciu k miestnosti. Na to využijem metódu `AttachClient2Room`, ktorej parametrami sú opäť identifikátory klientov, plus identifikátor miestnosti. V metóde vyvolám poslanie informácií o miestnostiach obom pripojeným aplikáciám.

Po prepojení klientských aplikácií a centrálneho bodu je možné posielať dáta o osobách stojacich pred senzorom (hudobníkoch) a interakcii s pripravenými hudobnými nástrojmi. Po prvotnej detekcii osoby pred Kinectom je nutné vyvolať metódu `NewMusician`, ktorá vytvorí registráciu hudobníka na centrálnom bode, pre danú miestnosť. Vďaka tejto registrácii je možné meniť meno a hudobný nástroj pre hudobníka priamo z centrálneho bodu a to pomocou volania metódy `ManageMusician`.

Vytváranie zvukového sprievodu vykoná klientská aplikácia volaním metódy `NewInteraction` so vstupnými parametrami, ktorými sú identifikátory centrálneho bodu a miestnosti a informácie o tom aký typ interakcie sa vykonal pomocou triedy `SynthInteraction`. Dáta interakcie pošlem ako spätné volanie `SendInteraction` do centrálneho bodu, kde na základe získaných informácií doplním zvukový sprievod.




*Obrázok 3.5: Sekvenčný diagram volaní niekoľkých metód medzi jednotlivými blokmi systému*

## 3.2 Centrálny bod

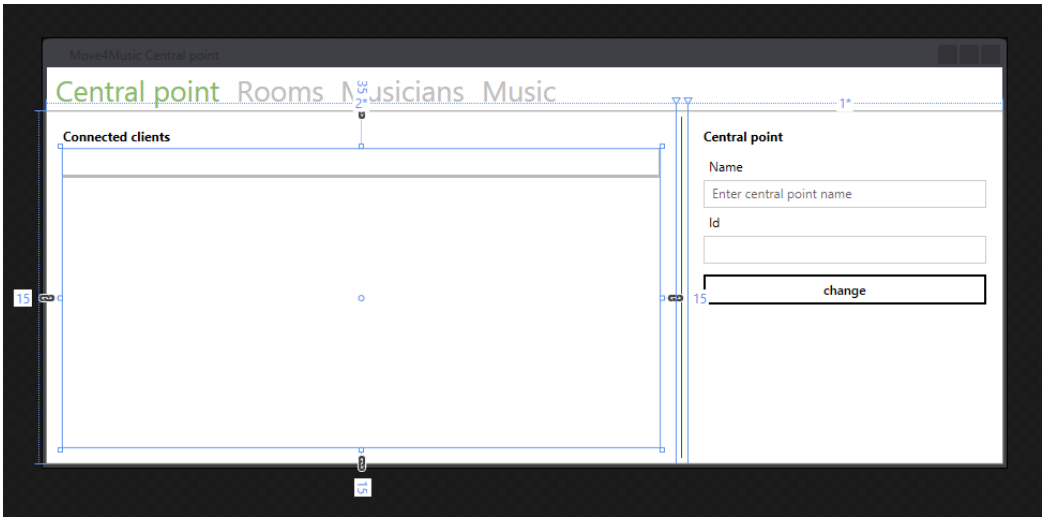
Aplikácia centrálneho bodu je WPF aplikáciou, ktorá sa stará o vytváranie zvukové sprievodu, pre niekoľko klientských aplikácií. Chcel som aby aplikácia pôsobila moderne a preto som na dizajn celkového vzhľadu využil voľne dostupné knižnice MahApps[11], ktoré upravujú štandardný vzhľad okna. Dizajn aplikácie sa tak priblížil modernému vzhľadu aplikácií v OS Windows 8/8.1. Na tvorbu zvukového sprievodu som v aplikácii centrálneho bodu použil knižnicu NAudio.

Okno aplikácie sa skladá zo štyroch záložiek (Obrázok 3.6):

- Centrálny bod – zobrazujú sa informácie o pripojených klientoch a centrálnom bode,
- Miestnosti – zoznam aktívnych miestností, ktoré klient poskytuje,
- Hudobníci – záložka zobrazuje zoznam hudobníkov, ktorý stoja v aktuálnej dobe pred senzorom,

[11] MahApps, http://mahapps.com/

- Hudba – záložka obsahuje ovládacie prvky na úpravu hlasitosti jednotlivých nástrojov, spustenie a zastavenie prehrávania hudby a konzolu s výstupom pre meranie komunikačných rýchlostí.




*Obrázok 3.6: Zobrazenie užívateľského rozhrania centrálneho bodu vo vývojovom prostredí*

Jednotlivé záložky tak neslúžia len na zobrazovanie informácií ale sú na nich umiestnené aj ovládacie prvky. Tieto zabezpečujú možnosť vytvárania zmien v aplikačných dátach, napr. zmena názvu, pridanie či odobranie miestnosti alebo zmena hudobného nástroja.

Všetky metódy, ktoré komunikujú so službou sú implementované ako asynchrónne. Zabezpečím tým plynulosť chodu aplikácie aj pri pomalej odozve v sieti. Pridaním referencie na WSDL súbor WCF služby, získam potrebné entity a metódy, ktoré budem na komunikáciu využívať.

V obidvoch klientských aplikáciách je nutné implementovať rozhranie `IPublishSubscribeAlphaCallback`.

Preto som vytvoril triedu `PublishSubscribeAlphaCallbackHandler`, v ktorej implementujem metódy pre spätné volanie a určím čo sa s prijatými dátami ďalej stane. Nakoľko všetky informácie využívam neskôr v rôznych miestach aplikácie, vytvoril som si pre každé spätné volanie udalosť, ktorá sa vyvolá pri prijatí callbacku z WCF služby. Keďže väčšina spätných volaní prináša so sebou nové dáta, pripravil som si generická triedu (Výpis kódu 3.3), ktorá dedí zo štandardnej triedy `EventArgs` a túto v týchto udalostiach používam. Ďalej som vytvoril generického delegáta pre všetky udalosti, na ktoré budem v aplikácii reagovať.

```
public delegate void GenericDataHandler<T>(object source, T e);
public class GenericDataEventArgs<T> : EventArgs
{
        public T Data { get; set; }
        public string Info { get; set; }
}
```

```
public event GenericDataHandler<GenericDataEventArgs<SynthMusician[]>> OnMusiciansReceived;
public void PublishMusicians(SynthMusician[] musicians)
{
        if (OnMusiciansReceived != null)
        {
                OnMusiciansReceived(
                        this,
                        new GenericDataEventArgs<SynthMusician[]>
                        {
                                Data = musicians
                        });
        }
}
```

*Výpis kódu 3.3: Ukážka vytvorenia a použitia generickej udalosti v aplikácii*

V aplikácie si zaregistrujem použitie udalostí, ktoré budem potrebovať. Po spustení aplikácie sa centrálny bod pokúsi o pripojenie k WCF službe. Asynchrónnym volaním metódy služby `RegisterCentralPoint` zaregistrujem klientskú aplikáciu ako centrálny bod. Následne čakám na odpoveď cez spätné volanie v udalosti `OnIdReceived`. Keď sa táto udalosť vyvolá, uložím si prijatý identifikátor do globálnej premennej a vyvolám založenie novej miestnosti volaním metódy `NewRoom`. Pre potreby diplomovej práce, je ako parameter metódy, názov miestnosti, využité náhodne vygenerované číslo. Takto pripravený centrálny bod čaká na spätné volania klientov, ktorý sa pripoja, a ich interakciu.

Po prijatí callbacku o novom hudobníkovi pre neho vytvorím model, inštanciu triedy `MusicianModel`, ktorá obsahuje prijaté dáta a informácie o miestnosti ku ktorej patrí. Tieto potom zobrazujem v zozname hudobníkov.

Metóda `OnInteractionReceived`, ktorá reaguje na udalosť slúži na vytvorenie zvukového sprievodu. Implementácia potrebných knižníc, ktoré poskytuje NAudio, sa nachádza v triede `AudioPlayer`. Táto trieda slúži na inicializáciu a ovládanie zvuku v celej aplikácii. Na prehrávanie zvukovej stopy používam štandardný výstup typu `WaveOut`, ktorý má implementované rozhranie `IWavePlayer` (Výpis kódu 3.4). Výstupom tak môže byť akákoľvek iná trieda, ktorá toto rozhranie implementuje. Je tak možné vytvoriť triedu na nahrávanie zvukového sprievodu do súboru.

```
private IWavePlayer _outputPlayer;
_outputPlayer = new WaveOut();
```

*Výpis kódu 3.4: Inicializácia štandardného výstupu pre NAudio*

Keďže je nutné aby bolo možné prehrávať niekoľko nástrojov vzájomne, na veľmi dlhú dobu, musel som upraviť triedu, ktorú NAudio na mixovanie používa. Vznikla tak trieda `ContinuousMixingSampleProvider`, ktorá dokáže zmiešať niekoľko vstupných súborov na jeden výstup, bez časového obmedzenia a bez obmedzenia vstupný súborov (Výpis kódu 3.5).

```
new ContinuousMixingSampleProvider(WaveFormat.CreateIeeeFloatWaveFormat(44100, 2));
```

*Výpis kódu 3.5: Inicializácia triedy na mixovanie zvukových vstupov*

Keďže použité výstupné zariadenie používa ako vstupný parameter typ `IWaveProvider` a trieda na miešanie zvukových vstupov implementuje rozhranie `ISampleProvider`, je nutné použiť triedu `SampleToWaveProvider`, na prevod. Potom môžeme použiť metódu `Init` (Výpis kódu 3.6) a inicializovať výstup pre zvukové zariadenie.

```
_sampleToWaveProvider = new SampleToWaveProvider(_mixer);
_outputPlayer.Init(_sampleToWaveProvider);
```

*Výpis kódu 3.6: Inicializácia triedy na mixovanie zvukových vstupov*

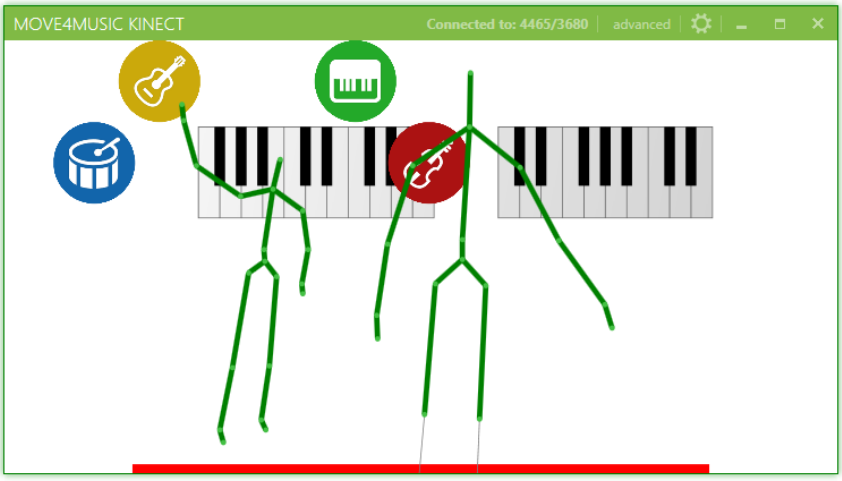
Trieda ďalej implementuje tri metódy pre zmenu stavu prehrávania hudby a to `Play`, `Pause`, `Stop`. Hlavnou metódou však je `PlayInteractionSound`, ktorej vstupným parametrom je získaná interakcia v podobe inštancie triedy `SynthInteraction`.

V dátach o interakcii si prenesiem názov súboru, ktorý sa má prehrať a podľa typu interakcie môžem určiť akým spôsobom. Na spracovanie interakcie slúži trieda `AudioMusician`. Táto trieda obsahuje metódy na načítanie a spracovanie zvukového súboru. Taktiež obsahuje cache, do ktorej si uložím už načítané zvukové súbory aby som obmedzil počet čítania z pevného disku. Pre niektoré typy interakcií je nutné opakovanie jedného zvukového súboru po dlhšiu dobu, než trvanie samotného zvuku. Pre toto použitie som implementoval metódu `HandleSound`, ktorá sa stará o vytvorenie opakovanej zvukovej slučky jedného súboru.

Využitím tried `AudioMusician` a `AudioPlayer` zabezpečujem hlavnú časť prehrávanie zvukových súborov. Zvukové súbory sú voľne dostupnými hudobnými vzorkami (z angl. samples) vo formáte .wav.

## 3.3 Klient pre Kinect

Pri tvorbe klientskej aplikácie som použil rovnako jednoduchý dizajn (taktiež pomocou knižníc MahApps, Obrázok 3.7) s maximálne využitým priestorom pre zobrazovanie výstupu zo senzora. Jedna klientská aplikácie ovláda jedno zariadenie Kinect. Pre viac zariadení pripojených na jednom počítači je možné spustiť viac inštancií aplikácie.



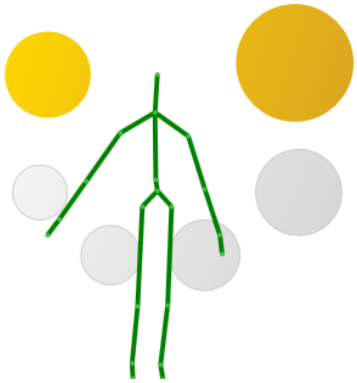

*Obrázok 3.7: Ukážka klientskej aplikácie*

### 3.3.1 Výber nástrojov

Pre implementáciu rôznych druhov hudobných nástrojov som využil rôzne druhy gest a pozícií užívateľa. Na základe dostupných zvukových vzoriek a snahy o odlíšenie druhov nástrojov som vybral hudobné nástroje: gitaru, husle, klavír a bicie.

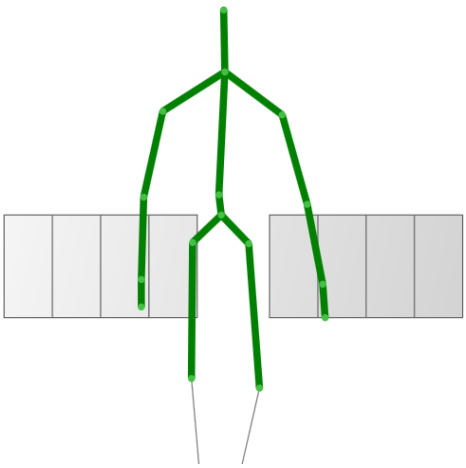
Všetky nástroje sú reprezentované jednoduchým zobrazením (štvorce alebo kruhy) a užívateľ musí vytvoriť nejakú interakciu práve v danom zobrazení.

Bicie nástroje – zostava kruhov, ktorá reprezentuje skutočné bicie nástroje. Interakcia na vytvorenie zvukového sprievodu vznikne pri vstupe rúk do vymedzených kruhov (Obrázok 3.8).



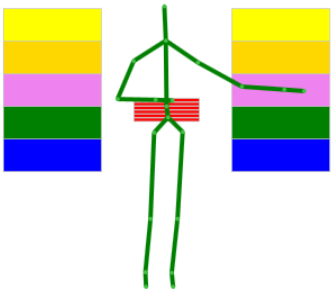
*Obrázok 3.8: Vizualizácia bicích nástrojov*

Klavír – predstavuje niekoľko kláves z klasického klavíru. Ide o niekoľko obdĺžnikov, kde každý z nich má priradený inú tóninu (Obrázok 3.9).



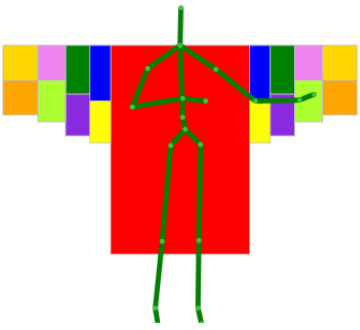
*Obrázok 3.9: Zobrazenie niekoľkých kláves na simulovanie klavíra*

Gitara – niekoľko obdĺžnikov, ktoré predstavujú tóniny na hmatníku a tenkých čiar, ktoré reprezentujú struny (Obrázok 3.10).



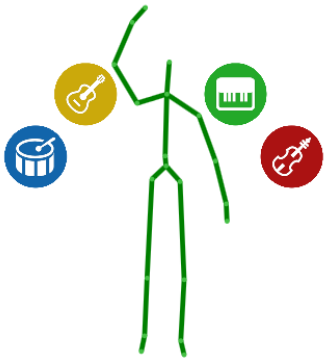
*Obrázok 3.10: Ukážka použitia zobrazenia nástroja ako gitary*

Husle – podobne ako gitara aj husle majú pre hmatník niekoľko obdĺžnikov, ktoré reprezentujú rôzne tóniny a jeden obdĺžnik na vytvorenie priestoru pre sláčik (Obrázok 3.11).



*Obrázok 3.11: Reprezentácia huslí*

Výber a zmena jednotlivých nástrojov (Obrázok 3.12) je možná z centrálneho bodu pre každého užívateľa osobitne alebo gestom zdvihnutia ruky nad hlavu v klientskej aplikácii. Po tomto geste sa pôvodný nástroj odstráni a zobrazí sa ponuka v podobe štyroch kruhov s ikonami nástrojov.



*Obrázok 3.12: Výber hudobných nástrojov pomocou gesta*

Tak ako centrálny bod, tak aj klientská aplikácia pre Kinect obsahuje referenciu na WCF službu. A podobne sú implementované aj potrebné rozhrania pre callback.

Pripojenie klienta k službe je taktiež implementované pri štarte aplikácie a podobným spôsobom ako v aplikácii centrálneho bodu aj tu je potrebné získať identifikátor, ktorý sa vráti v rámci spätného volania. Potom je možné pripojiť klienta k centrálnemu bodu, vďaka zoznamu, ktorý získame vyvolaním ponuky a následne po prijatí spätného volania. Po pripojení k centrálnemu bodu sa podobným spôsobom pripojíme k niektorej z dostupných miestností. V prípade, že klient nebude pripojený k centrálnemu bodu, zobrazuje sa o tom informácia v rohu aplikácie.

Okrem komunikácie so službou je nutné zabezpečiť aj výber aktívneho senzora. Na toto v aplikácii využijem komponentu `KinectSensorChooser`, ktorá je dostupná v knižnici `Microsoft.Kinect.Toolkit`.

Pridaním atribútu `xmlns:k=http://schemas.microsoft.com/kinect/2013` na element okna aplikácie zaregistrujem knižnicu pre použitie v XAML kóde.

```
<k:KinectSensorChooserUI Grid.Column="0" HorizontalAlignment="Center" VerticalAlignment="Top"
x:Name="SensorChooserUi" />
```

*Výpis kódu 3.7: Použitie komponenty na výber senzora vo WPF aplikácii*

Pridanie komponenty v XAML (Výpis kódu 3.7) vytvorí priestor pre zobrazenie stavu o pripojenom zariadení. Je potrebné však zabezpečiť výber a uloženie vybraného senzora. Na to je potrebné použiť triedu `KinectSensorChooser` a jej udalosť `KinectChanged`. V metóde, ktorá bude reagovať na zmenu senzora som pripravil nastavenie senzora a jeho uloženie do premennej pre ďalšie použitie.

Okrem komponenty na výber hudobných nástrojov sa v hlavnom okne nachádza komponenta `GesturesRegion`. Táto komponenta (Výpis kódu 3.8) sa stará o odchytávanie gest a zobrazovanie výstupu zo senzora. Poskytuje niekoľko udalostí, ktoré v hlavnom okne spracujem a vďaka ním tak komunikujem so službou WCF.

```
<kinectControls:GesturesRegion Grid.Column="0" x:Name="GesturesRegion"
GestureDetected="GesturesRegion_GestureDetected"
```

```
GestureNotDetected="GesturesRegion_GestureNotDetected"
PropertyChanged="GesturesRegion_OnPropertyChanged"
MusicianEnter="GesturesRegion_OnMusicianEnter"
MusicianLeave="GesturesRegion_OnMusicianLeave"
MusicianChanged="GesturesRegion_OnMusicianChanged"/>
```

*Výpis kódu 3.8: Komonenta GesturesRegion v XAML súbore*

Udalosti `GestureDetected` a `GestureNotDetected` sa vyvolajú vtedy, keď bola alebo nebolo zistené gesto. Druhú menovanú udalosť som využíval často iba v rámci testovania, vo výslednej aplikácii nie je používaná. Udalosti `MusicianEnter`, `MusicianLeave`, `MusicianChanged` slúžia na odchytenie zmeny, ktorá sa týka osoby stojacej pred senzorom. Viem tak určiť, kedy sa pred senzor postavila nová osoba alebo odišla, či zmenila nejakú zo svojich hodnôt (využíva sa na zmenu hudobného nástroja).

Táto komponenta okrem kompletného spravovania užívateľov a práce s jednotlivými gestami sa stará aj o vykreslenie výstupu zo senzora.

```
_sensor.ColorStream.Enable();
_sensor.SkeletonStream.Enable();
_sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
```

*Výpis kódu 3.9: Spustenie prúdov v aplikácii*

Senzor využíva tri druhy vstupných dátových tokov (Výpis kódu 3.9). Pre video vstup je to štandardné nastavenia `RgbResolution640x480Fps30`, ktoré nie je potrebné uvádzať. V metóde, ktorá reaguje na udalosť `AllFramesReady` potom tieto dáta spracujem. Primárne využitie v aplikácii má `SkeletonStream`, vďaka ktorému je následne aj vykreslená fiktívna kostra osoby spolu so zobrazením bodov, ktoré senzor sníma (joints).

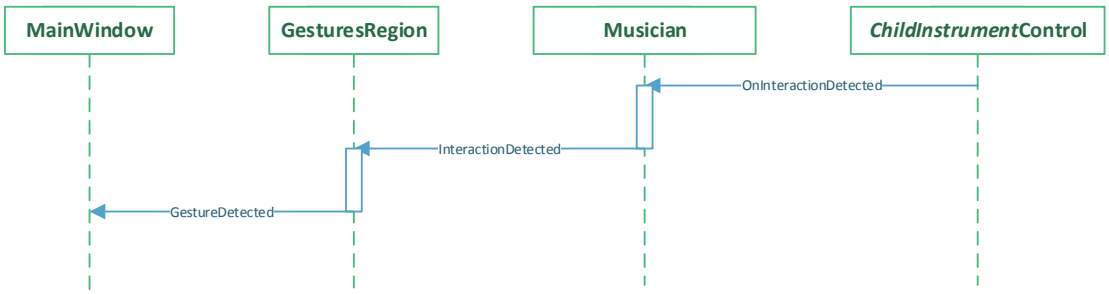
Okrem osoby sa na výstup vykresľujú aj jednotlivé nástroje, ktoré zmenou pozície a svojej veľkosti sledujú pohyb užívateľa. Každého užívateľa, ktorý stojí pred senzorom predstavuje inštancia triedy `Musician`, ktorá si vo svojich vlastnostiach uchováva informácii o danej kostre, názve a hudobnom nástroji.

Všetky hudobné nástroje dedia z komponenty `InstrumentControl`. Táto komponenta sa samostatne nepoužíva, no slúži na vytvorenie všetkých ostatných komponent hudobných nástrojov. Táto trieda taktiež obsahuje niekoľko užitočných metód na prepočet pozícií a veľkostí, či na detekciu niektorých jednoduchých gest. Tieto metódy sú potom využívané v zdedených komponentách.

Každý nástroj definuje v XAML súbore vizuálne zobrazenie nástroja a na strane riadeného kódu sú implementované metódy, ktoré zabezpečujú relatívne premiestňovanie a zmenu veľkosti tohto zobrazenia. Taktiež som tu implementoval logiku detekcie gest, ktorá je špecifická pre každý nástroj. Ak teda pozícia kostry odpovedá podmienkam, ktoré sú implementované pre daný nástroj (napr. keď sa bod pozície ruky nachádza v oblasti štvorca, ktorý predstavuje časť hudobného nástroja), vyvolám udalosť s predaním potrebných parametrov (Výpis kódu 3.10).

```
OnInteractionDetected(new InteractionDetectedEventArgs
{
        InstrumentType = InstrumentType.NewGuitar,
        InteractionDetectedAt = DateTime.Now,
        InteractionType = InteractionType.OnEnter,
        SoundInfo = new SoundInfo
        {
                Name = soundName,
                Path = "/Guitar/",
                Volume = 1f
        },
        Sensor = Sensor,
        Skeleton = Skeleton,
        IsContinous = false
});
```

*Výpis kódu 3.10: Vyvolanie udalosti interakcie v komponente pre hudobný nástroj (gitara)*




*Obrázok 3.13: Postup vyvolania udalosti na detekciu gesta*

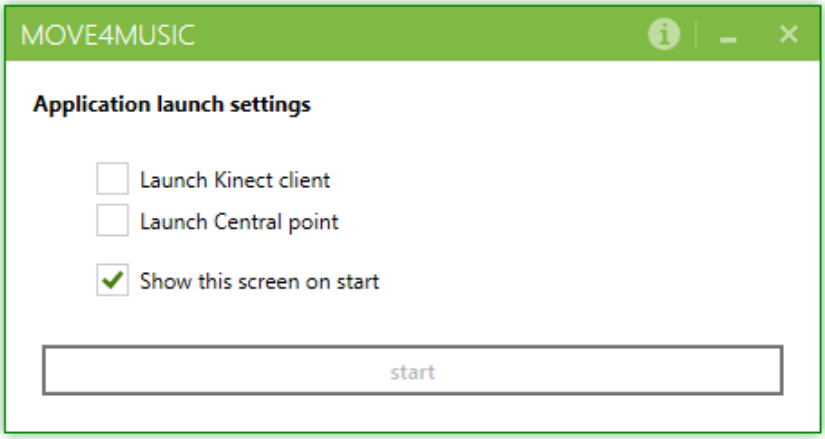
Vyvolanú udalosť na jednotlivých miestach v programe ešte spracujem a zhodnotím. Nakoniec sa vyvolá udalosť `GestureDetected`, v ktorej asynchrónne zavolám metódu `NewInteraction` vo WCF službe spoločne s potrebnými získanými parametrami na vyvolanie interakcie na strane centrálneho bodu (Obrázok 3.13).

## 3.4 Nasadenie aplikácie

Na vytváranie verzií aplikácie a následnú inštaláciu na klientské stanice som využil technológiu ClickOnce [12], spoločnosti Microsoft. Táto technológie podporuje jednoduchú inštaláciu aplikácií jednoduchým kliknutím na vystavený odkaz a ich následnú jednoduchú aktualizáciu. Odkaz na inštalačný súbor aplikácie[13] som taktiež umiestnil na server služby Azure.

[12] ClickOnce, https://msdn.microsoft.com/en-us/library/142dbbz4(v=vs.90).aspx
[13] Move4Music, http://granak.azurewebsites.net/m4m/publish.htm

Keďže systém obsahuje dva oddelené bloky klientských aplikácií, ktoré ale môžu byť spustené na jednom počítači, vytvoril som jednotnú spoločnú spúšťaciu aplikáciu (Obrázok 3.14). Túto aplikáciu tvorí jednoduchý dialóg so zaškrtávacími položkami pre možnosť spustenia jednotlivých klientských aplikácií.

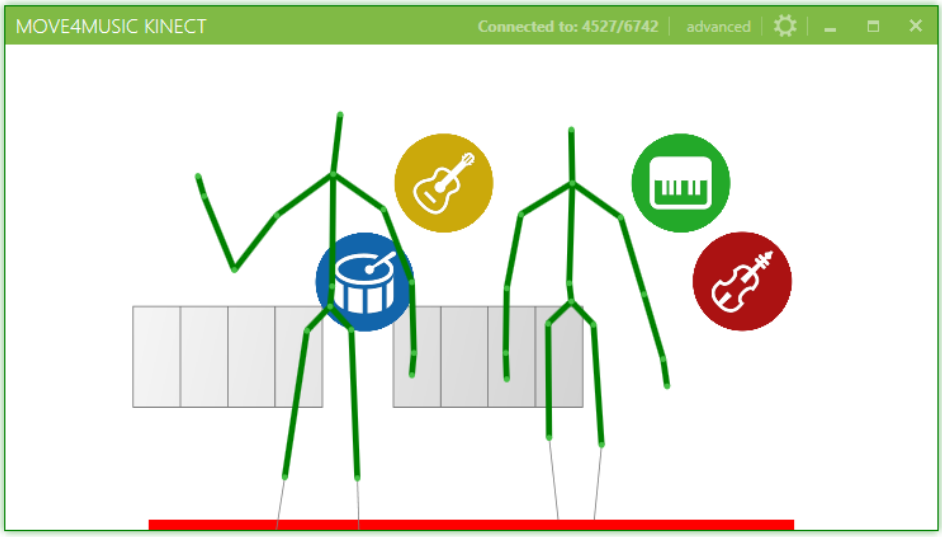



*Obrázok 3.14: Prvá obrazovka aplikácie*

# 4 Testovanie

Testy aplikácie som robil neustále popri jej vývoj. Boli to jednoduché testy na odozvu detekcie gesta v rámci klientskej aplikácie, no taktiež aj testovanie kompletného prenosu dát medzi jednotlivými časťami systému. Výsledkom testovaní prenosu dát je rozdiel času, ktorý bol zaznamenaný pri vykonaní akcie na strane klientskej aplikácie a času, ktorý bol zaznamenaný pri príchode dát do centrálneho bodu.

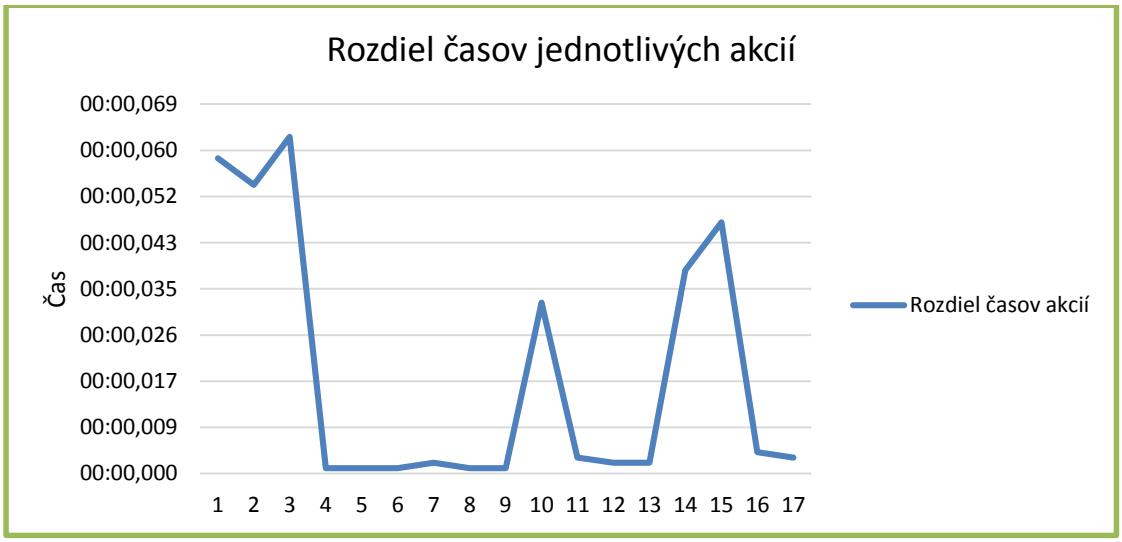
Testovanie gest som robil ich opakovaním za pomoci jednoduchého textu do konzoly vývojového nástroja. Tak som mohol overiť za akých okolností bolo gesto rozpoznané a prípadné problémy odstrániť. Na opakované testovanie ovládanie aplikácie a hudobných nástrojov senzorom, som používal aplikáciu Kinect Studio, ktorá je súčasťou SDK. Tá zabezpečí jednoduchší vývoj a testovanie detekcie gest vďaka možnosti nahrávania výstupu zo senzora a jeho opätovného využitia. Taktiež som overil možnosti s nastavením, keď senzor snímal dvoch užívateľov zároveň. V prípade menšieho zobrazenia aplikácie sa hudobné nástroje používateľov prekrývajú, no interakcia vždy patrí danej osobe (Obrázok 4.1).




*Obrázok 4.1: Testovanie aplikácie s dvoma osobami súčasne*

Na testovanie odozvy aplikácie som klientské časti systému upravil a pridal tak možnosť zobrazenia výstupu s časovou známkou a vykonanou akciou. V aplikácii centrálneho bodu je možné zobraziť tieto informácie prepnutím aplikácie do rozšíreného módu, kde sa na záložke hudby zobrazí panel s podrobným výpisom. V klientskej aplikácii sa tento panel zobrazí na pravej strane aplikácie, zmenšením priestoru pre vizualizáciu osôb a nástrojov.

## 4.1 Výsledky meraní oneskorenia prenosu dát



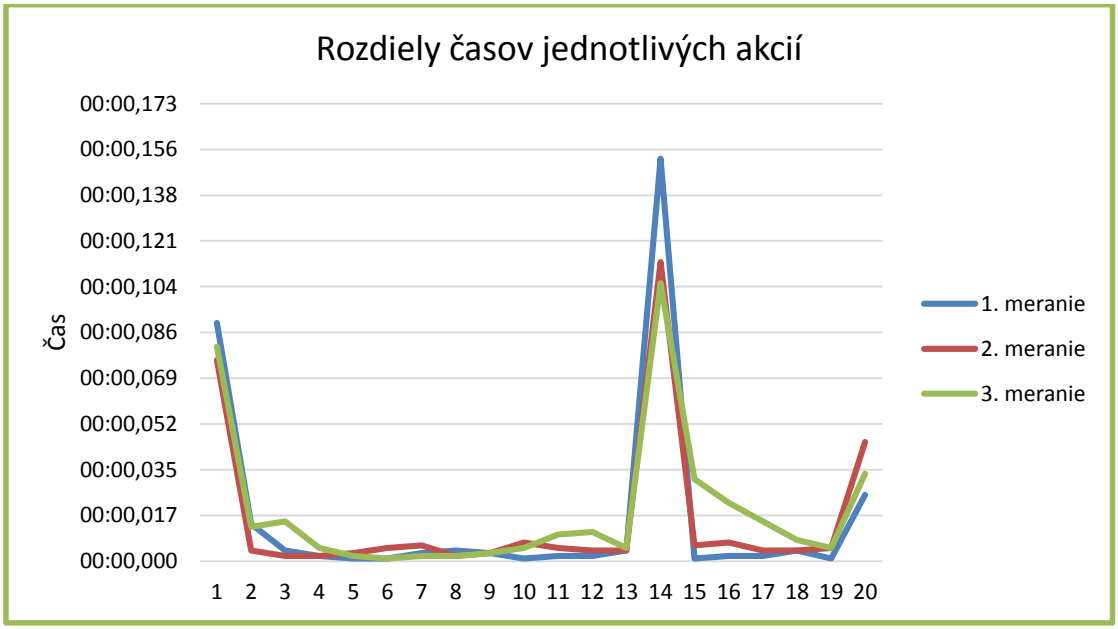

*Graf 4.1: Oneskorenie komunikácie na jednom PC*

Testy v rámci vývoja aplikácie som vykonával prevažne na jednom zariadení, no hlavne s využitím WCF služby umiestnenej v službe Microsoft Azure.

| | Čas (mm:ss,000) | Akcia | Popis akcie | Akciu vykonal |
|---|---|---|---|---|
| **1** | 00:26,735 | Musician entered | Senzor zaregistroval osobu | Klient |
| **2** | 00:26,794 | Room data received | Prijaté dáta na centrálny bod | Centrálny bod |
| **3** | 00:27,914 | Musician changed | Osoba si vybrala hudobný nástroj | Klient |
| **4** | 00:27,968 | Room data received | Prijaté dáta na centrálny bod | Centrálny bod |
| **5** | 00:28,351 | Musician entered | Druhá osoba vstúpila pred senzor | Klient |
| **6** | 00:28,414 | Room data received | Prijaté dáta na centrálny bod | Centrálny bod |
| **7** | 02:47,746 | Musician changed | Druhá osoba si zvolila nástroj | Klient |
| **8** | 02:47,747 | Room data received | Prijaté dáta na centrálny bod | Centrálny bod |
| **9** | 03:34,592 | Gesture detected | Zistené gesto | Klient |
| **10** | 03:34,593 | Interaction data received | Prijaté informácie o geste | Centrálny bod |
| **...** | ... | ... | ... | ... |
| **29** | 03:48,313 | Gesture detected | | Klient |
| **30** | 03:48,360 | Interaction data received | | |
| **31** | 03:55,116 | Musician leaved | Osoba opustila priestor senzoru | Klient |
| **32** | 03:55,120 | Room data received | | Centrálny bod |
| **33** | 03:55,125 | Musician leaved | Druhá osoba opustila priestor senzoru | Klient |
| **34** | 03:55,128 | Room data received | | Centrálny bod |

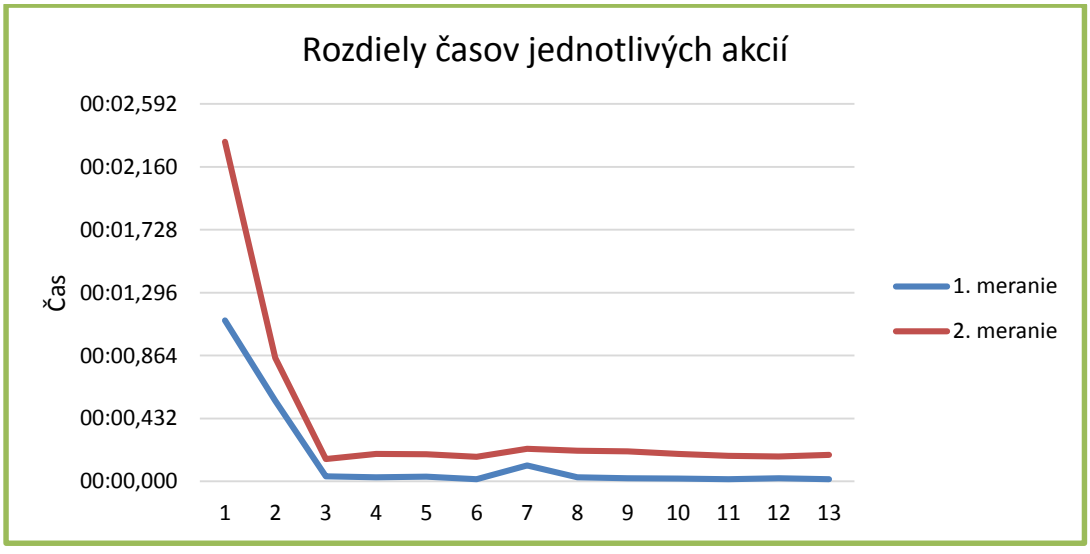
*Tabuľka 4.1: Meranie času oneskorenia komunikácie; jeden PC, dvaja používatelia*

Pre vysokorýchlostné pripojenie cez LAN sa časové rozdiely pohybujú v rozmedzí tisícin až stotín sekundy (Tabuľka 4.1, Graf 4.1). Vďaka tomu je odpoveď centrálneho bodu vždy plynulá a bez akéhokoľvek väčšieho oneskorenia.




*Graf 4.2: Oneskorenie v komunikácii dvoch PC; jeden používateľ*

Z Graf 4.2 je možné vidieť, že najväčšie časové rozdiely v prenose dát sú pri príchode užívateľa pred senzor (bod 1) a následne majú hodnoty klesajúcu tendenciu. Bod 14 predstavujú miesto, kde si užívateľ zvolil iný hudobný nástroj. Spomalenie je zrejme zapríčinené potrebnou aplikačnou logikou na zmenu nástroja. Každé meranie zobrazené v Graf 4.2 bolo vykonané v rôznom čase (rozdiel v dňoch), no s rovnakým postupom akcií.




*Graf 4.3: Oneskorenie v komunikácii cez bezdrôtovú sieť*

Pre ďalšie meranie som využil bezdrôtovú sieť spoločne s testovaním zmeny konfigurácie spojenia. Pri testoch bezdrôtovej siete som narazil na obmedzenie politiky v sieti, kde nebola povolená komunikácia za použitia spojenia cez websocket. 1. meranie v Graf 4.3 predstavuje testovanie v rámci konfigurácie cez TCP spojenie avšak rýchle pripojenie k internetu. 2. meranie je spojenie cez websocket, no pomalšie pripojenie k internetu.

Výsledkom týchto testov je overenie možnosti systému vytvorenia zvykovej odozvy na vytvorené gesto užívateľom. Ako je vidieť z vyššie uvedených grafov, systém je schopný vytvárať zvukový sprievod v reálnom čase si minimálnom odozvou.

# 5 Záver

Senzor Kinect poskytuje množstvo užitočných dát, ktoré sa vďaka rôznym spôsobom spracovania stávajú prínosnými informáciami pre vývoj NUI a otvárajú veľa možností pre technologický rozvoj. Kinect pre Windows a jeho SDK poskytuje vývojárom aplikácií výborné rozhranie pre prístup k dátovým tokom a ovládaniu senzora. Vďaka tomu je vývoj aplikácií pre Kinect rýchly a zároveň aj na vysokej technologickej úrovni.

V spojení s technológiou WCF, ktorá je veľmi silným nástrojom pre sieťovú komunikáciu a SOA aplikácie, bolo možné vytvoriť systém hudobného sprievodu pre senzor Kinect. Systém tvorí základná kostra, ktorá predstavuje prepojenie klientských aplikácií so službou cez publish-subscribe prístup. Samotné klientské aplikácie sú vytvorené pomocou modernej technológie WPF, ktorá je základom pre univerzálne aplikácie, ktoré predstavujú budúcnosť rozvoja operačného systému Windows. Aj vďaka tomu je možné systém naďalej jednoducho a rýchlo rozširovať o nové vlastnosti a funkcie.

WCF službu je možné použiť ako univerzálne API, pre rýchlu komunikáciu bez nutnosti použitia senzora Kinect. Môže tak slúžiť ako serverová aplikácie pre odosielanie rôznych správ, či ovládanie vzdialených aplikácií.

Systém je rozšíriteľný aj z hľadiska doplnenia ďalších hudobných nástrojov, či implementácie rozpoznávania nových gest. Taktiež je v dobe písanie práce k dispozícii aj nová verzia senzoru Kinect, ktorý je možné v ďalšom vývoji systému implementovať. Ďalším vývojom klientských aplikácií môže byť podpora nových univerzálnych aplikácií pre OS Windows, webových alebo aj mobilných aplikácií.

# 6 Použitá literatúra


[1] VISUAL TARGET TRACKING. *Espacenet* [online]. [cit. 2015-04-07]. Dostupné z: http://worldwide.espacenet.com/publicationDetails/biblio?CC=WO&NR=2010088032A2&KC=A2&FT=D&date=20100805&DB=EPODOC

[2] E3 2010: Project Natal is "Kinect". *Ign.com* [online]. [cit. 2015-04-07]. Dostupné z: http://www.ign.com/articles/2010/06/13/e3-2010-project-natal-is-kinect

[3] WEBB, Jarrett a James ASHLEY. 2012. *Beginning Kinect programming with the Microsoft Kinect SDK*. New York: Apress, xvii, 306 p. Expert's voice in Microsoft. ISBN 14-302-4104-7.

[4] Kinect. 2015. *Wikipedia* [online]. [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Kinect

[5] Natural user interface. 2015. *Wikipedia* [online]. [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Natural_user_interface

[6] MICROSOFT. 2015. *Kinect for Windows SDK* [online]. [cit. 2015-05-06]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh855347.aspx

[7] What Is Windows Communication Foundation. 2015. *MSDN* [online]. [cit. 2015-05-06]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx

[8] WCF Essentials: What You Need To Know About One-Way Calls, Callbacks, And Events. 2015. LOWY, Juval. *MSDN Magazine* [online]. [cit. 2015-05-06]. Dostupné z: https://msdn.microsoft.com/en-us/magazine/cc163537.aspx

[9] Task-based Asynchronous Pattern (TAP). 2015. *MSDN* [online]. [cit. 2015-05-06]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh873175(v=vs.110).aspx